Departamento de Tratamiento de la Señal y Comunicaciones

TRABAJO FIN DE GRADO

Extracción de descriptores de movimiento en vídeos para la evaluación de la estética

Autor: Paloma Tirado Martín

Tutor: Alejandro Hernández García

Leganés, Octubre 2015

Título: Extracción de descriptores de movimiento en vídeos para la evaluación de la estética
Autor: Paloma Tirado Martín
Tutor: Alejandro Hernández García

EL TRIBUNAL

Presidente: Francisco Javier González Serrano

Vocal: Pablo Basanta Val

Secretario: Francisco Javier Herráiz Martínez

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día 15 de Octubre de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

"Toda la vida he tenido un sueño: conseguir todos mis objetivos."

Homer J. Simpson

Agradecimientos

En primer lugar, agradecer a mi tutor Alejandro sus consejos y paciencia, así como su esfuerzo por estar siempre disponible, sobretodo cuando se encontraba correos demasiado largos y llenos de pánico en pleno mes de agosto.

Es necesario mencionar a Irene, Sara, Elena y Laura, por mejorar esos fines de semana interminables de biblioteca y las prácticas hasta las 9 de la noche, que han sido muchas durante estos años. No puedo olvidarme de Lidia, porque a pesar de la carencia de coordinación de nuestros calendarios, nunca ha dejado de estar ahí aunque fuese en la distancia.

A Juan, porque tuvo la mala suerte de tener que soportarme durante la época más difícil y más quejica de la carrera, y aún así nunca dejó de hacerlo.

Gracias a mi hermano Juan Manuel, por ayudarme a lo largo de estos años en todo lo que podía, incluso desde la distancia, y conseguir la gran hazaña de que no llegara a odiar la programación.

Finalmente, y no por ello menos importante, gracias mis padres Manuel y María Luisa, porque sin su dedicación y apoyo durante estos años, esto no habría sido posible. Gracias por creer en mí cuando ni yo misma lo hacía y hacer posible que hoy esté escribiendo estas líneas. El mérito es más vuestro que mío.

Summary

The growth of video streaming has increased noticeably through the last decade. Because of it, the task of searching and recommending videos is becoming more and more difficult. Whereas before the existence of video streaming information retrievals was only based on text and metadata, nowadays content-based image and video retrieval is starting to be researched. In order to add value and success to user's searchings, it is interesting to assess the quality and aesthetic value of the information it is retrieved.

On this thesis we are going to extract several motion related descriptors in order to aesthetically assess a car commercial database. The videos in the database are extracted from *YouTube* and labeled in order to metadata provided by the website. Specifically three kinds of labeling are going to be used: based on quality or likes/-dislikes, quantity or number of views and the combination of both of them. Quality and quantity provide a binary labeling, and the combination clusters the videos in four classes.

As it is usually done in computer vision, the main objective is suggesting a set of descriptors and designing and providing the procedures for calculating their values on the corpus of videos. These values are called descriptors, and they can be obtained by processing frames and handling the data got on the procedure to get specific numbers. With their help it may be possible to know whether they give enough information to determine the aesthetic appealing of the videos. On this project we are going focus on motion descriptors.

As an approach to get data about the video motion, the optical flow is estimated between each pair of frames. To do so, a *Matlab* friendly *C++* code developed in [16] is used. This algorithm is based on the brightness constancy assumption between two frames, leading to a continuous spatial-temporal function. This is discretized, linea-

rized and the temporal factor removed by assuming only the function in two frames. Afterwards the zero gradient values are found using *Iterative Reweighted Least Squares (IRLS)*, a method which iterates calculating different weights in order to find the ones fulfilling the zero gradient condition. From this a linear system is obtained and it is solved by using *Successive Over-Relaxation (SOR)* method, which is Gauss' variant with faster convergence.

The optical flow algorithm needs several parameters to be set. Because of the difficulty of setting these parameters automatically, these values are determined by the observation of each performance and efficiency representing the observed motion. When the optical flow is calculated, we filter homogeneous texture regions, due to possible error estimation induced by similar pixel values on the neighborhood. In order to determine the texture level on different frame regions, we measure the entropy on each one, which will provide a measurement of pixel's randomness. This is made by turning each frame into gray-scale and dividing them into 60 different windows. Afterwards, a threshold is set to determine which region will be considered as a low texture one. This is done considering that filtering excessively could mean that the descriptors extraction will not be representative. However, in cases with a lot of very homogeneous regions (e.g.: completely black) the amount of vectors discarded will be high no matter what threshold is set. Then, when the region's entropy is less than the threshold, it will be considered as a low texture one and, as a consequence, its optical flow vectors will not be taken into consideration.

After filtering based on the texture, the first step is getting the angle and modulus of the movement estimated in every pixel using the components got. For easy direction interpretation when getting the different descriptors, the angles are nominalized according to the 8 cardinal points.

By using both cardinals and modulus obtained, it is possible to estimate approximately which camera motion is taking place on every frame or shot. For this we are only taking into account those values on the margins of each frame. Previous to the camera motion detection, it is necessary to apply some weights to the cardinal values on the margins. This is done not only to give relevance to N-S-E-W

cardinals, but also to diagonal ones, because they give information about the camera motion although they do not belong to the purest pan and tilt motion types. Adding each different weight depending on the cardinal, we get a percentage in comparison to the ideal motion type (this is, every pixel moving towards the same direction) which gives out the "amount" of movement going to each N-S-E-W direction.

The most common shot type on the database is done by using fixed cameras and it is detected by setting a threshold to the mean modulus of the margins of each frame, which should include those frames which are fixed but have some kind of movement on the margins because of the captured scene. If it is less than the mentioned threshold, the frame will be considered as fixed. If not, we begin to detect zoom presence. In order to do that, margins are divided into 2 vertical and 2 horizontal regions, and each maximum percentage cardinal is obtained. When detecting which type of zoom it is, we know the specific directions each margin should have in theory. Starting from that, we can compare the theoretical value with the maximum direction got before using weights. When 3 or 4 of the margin's directions correspond to the theoretical pixels motion in each zoom type, the current frame will be considered as one with zoom in or out, depending on the conditions. Considering the zoom just under 3 conditions is not that restrictive, and this is because having just those 3 conditions fulfilled is very unlikely unless we have a zoom. If the zoom is not the case, we evaluate whether it is a pan or tilt camera motion. Now, the maximum percentage is obtained among all the margins instead of dividing them into regions, because directions should be the same along all the frame. If the difference of the maximum value with respect to the rest of the cardinal percentages is greater than a threshold, and the maximum value is higher than another different threshold, it will be considered pan right/left or tilt up/down depending on the cardinal which belongs to the maximum. This is done in order to discard those maximums in non predominant directions. Finally, if none of these conditions are fulfilled, the frame will have assigned a non-specific motion.

Once frame camera motion is determined, we proceed to detect shots on each video by computing the *Sum of Absolute Differences*

(SAD) of the gray intensity pixels and its first and second derivatives. Shot changes are detected when this second derivative has a value greater than a chosen threshold. After shot detection, the mode camera motion type is obtained, as well as the percentage of frames on the shot with that value. When this percentage is greater than a threshold, it is considered the shot has a predominant value which corresponds to the most present motion on the frames.

At this point the data computed are not at video level. This is why we need to use the data in order to obtain single values which represent each video. In order to get statistical parameters at video level, it is not possible to iterate creating a matrix with every single angle or modulus value, because it is highly memory and computing time consuming, so it is necessary to do it sequentially. This means getting single values on each frame which will be helpful when computing statistical video descriptors. As angles have a circular nature, using circular statistics is compulsory. For this purpose we are going to store only the sum of each vector component through the whole set of frames, as well as the sum of modulus. We also obtain the number of pixels taken into account on the operation because it will not be constant due to the low texture region filtering. With these values we get everything needed to compute the mean an standard deviation at video level. However, when dealing with data like camera motion type through frames and shots, it is not possible to get means and standard deviations, and this is why we get the percentage of each motion type at shot and frame level.

When data handling is done, we extract 27 different descriptors which are going to be evaluated using the three labeling methods previously referred. By using machine learning algorithms provided by *Weka*, several features and classifiers are tested, getting the best performance using quantity labeling and angle and modulus related features, getting a 60 % accuracy with tree classifier *SimpleCart*. Although in general descriptors performance is not remarkably good, by using the *Experimenter* tool provided by *Weka*, we can find out which set of features and classifiers really provide a statistically significant improvement with respect to *ZeroR* classifier. We observe that with combination labeling the accuracy is less than those in quality and quantity labeling. This is because combination deals

with 4 different types of labels, meanwhile quality and quantity are binary, although it does not mean that combination labeling works worse than the binary ones, because its improvement with respect to *ZeroR* could be better. In fact, combination labeling gives more information because it has an statistically relevant performance when choosing angles and modulus related features and *SimpleCart* and *SimpleLogistic* classifiers, which means that the accuracy percentage is not something casual. We also get significant results when using quantity labeling and the same set of features with *SimpleCart*.

These results lead to the conclusion that camera motion is not particularly relevant when assessing aesthetics on this database. This is something contradictory to what one might think, because camera motion is used typically to add drama on an audiovisual context. An explanation to this could be the fact that in general, fixed and hand-carried camera motion are noticeably common on the database and that is why it does not really affect when the user decides whether he likes it or not. In addition, it is well known that establishing ground-truth when dealing with people's likings is not trivial because of their subjectivity, and this could affect results. The lack of ca database with camera motion labels is also crucial, because it makes difficult knowing if the rest of the non-manually labeled videos behave correctly when the camera motion detection method is applied.

In this project we check that not always theoretical knowledge corresponds to what it is observed in a practical context, but we also provide a way to extract descriptors and analyze them with a simple approach. This could be improved in the future by labeling the database with respect to the camera motion and by segmenting background in order to improve the steady camera detection. Binary aesthetic labeling could be also improved by using supervised annotation extracted by measuring involuntary biological responses experienced by the evaluator.

Keywords: optical flow, machine learning, camera motion, aesthetics assessment.

Resumen

En este proyecto se extraen diferentes características relacionadas con el movimiento de los vídeos proporcionados en la base de datos, los cuales se corresponden con anuncios de coches. Para ello, se proporciona una estimación del flujo óptico haciendo uso del algoritmo proporcionado en [16].

Mediante un análisis del flujo óptico en los márgenes de los fotogramas se caracteriza el movimiento de cámara presente en éstos y se calculan los ángulos y módulos correspondientes al movimiento de cada píxel. Posteriormente se procede a un cálculo secuencial de estos valores para obtener descriptores a nivel de vídeo.

Con los datos obtenidos y con tres diferentes etiquetados de los vídeos basados en calidad, cantidad y en la combinación de éstos, se aplican métodos de aprendizaje máquina con diferentes conjuntos de descriptores y clasificadores para la evaluación de la estética, la cual estará basada en los metadatos proporcionados por los usuarios a través de *YouTube*. Se concluye de esta manera que el tipo de movimiento de cámara no afecta notablemente a la evaluación estética por parte de los usuarios, mientras que sí lo hacen el ángulo y módulo presentes en cada vídeo.

Palabras clave: flujo óptico, aprendizaje máquina, movimiento de cámara, evaluación de la estética.

Índice general

Índice de figuras

Índice de tablas

1. Introduction

1.1. Project motivation

Video streaming services' pupularity has grown rapidly in the last decade. This is due to the rise of services such as *YouTube* and *Vimeo*, which allow uploading and watching videos without cost and at any moment, providing social network services through the interaction among users and the possibility of giving feedback about the public content.

Thanks to this, the amount of stored videos on the Internet has experienced a fast increase. This is why the task of carrying out a successful search through a database such as *YouTube*'s is getting more and more difficult because of its dimension. With the aim of improving user's experiences when getting the expected result on their searchings, video classification and recommendation have become a much studied field recent years.

This project is an extension of that work in [7] in which a database based on car commercials is provided, labeled according to their aesthetics or consumer's perception and proposes a set of visual descriptors. This project will focused, particularly, on the analysis of motion-related descriptors, using the same database. This is because the utilization of movement to catch spectator's attention and emphasize drama is very common in film and commercial production and assessing these visual features might provide relevant information about consumers video perception.

Accomplishing an assessment about which motion features and how they affect when users watch the video will improve the platform's recommendation and searching performance. This is because it will be able to only retrieve those videos automatically assessed as quality ones. This also contributes to fields such as marketing,

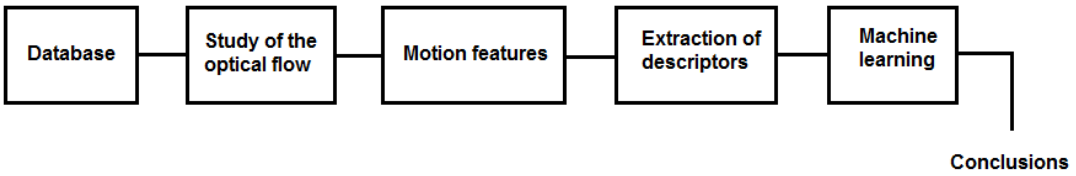


Figura 1.1: Steps followed on the project

making possible to estimate whether users will like the commercials or not, even before being released.

This project's aim is extracting several motion features for each video in the database to subsequently analyze the features' influence on users' general video perception.

1.2. Document structure

In chapter 2, which refers to the state of art, some basic and essential concepts related to this work will be explained in order to improve understanding, as well as different previous techniques developed to solve some of the addressed problems. Once finished, we will proceed to present the main development of the project, being every step needed explained on the figure 1.1.

Chapter 3 explains briefly those steps followed to choose and label the used video database, to continue with chapter 4, which deals with the optical flow. This section will explain how the chosen optical flow algorithm works, as well as the relevance of each one of its parameters and the procedure to set them.

After analyzing the optical flow, the extraction of different motion features will be detailed in chapter 5: angles, modulus and camera motion. The method applied to estimate the last one will also be specified in this chapter.

Once the features are obtained, we will proceed to extract each descriptor in chapter 6 and analyze their behavior in chapter 7 to continue explaining the project's management in chapter 8.

Finally, chapter 9 will close the document through a conclusion and several suggestions for possible future developments related to this project.

2. Estado del arte

En este apartado se hará referencia a aquellos conceptos y trabajos realizados previamente en relación con éste, así como los que sirvieron de ayuda para llevar a cabo cada uno de los pasos realizados.

2.1. Computer vision

Debido al desconocimiento del funcionamiento del sistema visual en los animales, se desarrolla un nuevo campo científico llamado *computer vision* o visión artificial. Según se explica en [27], mientras que los humanos partimos de un objeto ya existente, el cual percibimos con todo tipo de detalle, los investigadores de *computer vision* tienen como objetivo recrear éstos de manera fiable utilizando diversos métodos matemáticos.

La dificultad de esta tarea reside en que pretende dar una solución a un problema del que se parte con información insuficiente y con la cual se ha de llegar a un resultado específico. Para compensar estas carencias, el campo de *computer vision* se sirve de modelos basados en la física y la probabilística. La visión artificial está altamente relacionada con el análisis de imagen o *imagen analysis* así como del procesado de imagen o *image processing*. Ésto campos pueden aplicarse a otros como la neurobiología, la física, la inteligencia artificial o el procesado de señales.

2.1.1. Descriptores

Para entender aquellos trabajos realizados en el campo de la visión artificial, es necesario conocer el concepto de descriptor. Un descriptor visual ofrece información sobre algunas de las múltiples

características visuales presentes en vídeos o imágenes. Con la extracción de éstos se pueden establecer relaciones entre los valores de los píxeles y la percepción humana. Sin embargo, debido a que ésta es subjetiva y depende del usuario que observa, los resultados serán aproximaciones.

Dentro de los descriptores, se pueden encontrar aquellos de "bajo nivel", como el color, la textura, el brillo, la forma o el movimiento. Con éstos podemos obtener descripciones sobre el movimiento de cámara, el tipo de formas que existen o cómo de homogéneos son la imagen o vídeos analizados. Existen también descriptores de "alto nivel", los cuales aportan información sobre objetos y sucesos que ocurren en la imagen o vídeo, como la detección de caras.

2.1.2. Funciones y aplicaciones

Gracias al computer vision se pueden llevar a cabo tareas de análisis de movimiento y de reconocimiento, que llevan a cabo diversas funciones. Estas son algunas de ellas:

- Odometría visual: o también llamada *visual odometry* que, llevada a un nivel 3D, se encarga de determinar el movimiento de cámara relativo con respecto a una escena. Esta técnica se usa, por ejemplo, en el movimiento de robots autónomos, como los implicados en labores de exploración espacial.

- Flujo óptico: también llamado *optical flow*, es el aparente patrón de movimiento de una escena, ya sea de los objetos presentes en ésta o de sus bordes, que es provocado por el movimiento respecto del ojo o cámara que lo captura. La estimación del flujo óptico se puede utilizar para tareas como el análisis del tráfico, utilizando el flujo óptico para la segmentación en clusters de la imagen [24]. Este concepto se explicará en más profundidad en el apartado 2.2.

- Seguimiento: o también llamado *tracking*. Localiza el movimiento de un objeto o región concreta dentro de una imagen o vídeo. Se puede aplicar para la interacción de humanos con ordenadores o para seguridad y vigilancia, detectando así movimientos indeseados.

- Reconocimiento facial: identifica o verifica personas mediante la extracción de características en una imagen o vídeo, y comparándola con otras buscando correspondencias. Esta técnica es la utilizada en los pasaportes digitales para utilizar una verificación automática, prescindiendo así de oficiales que las lleven a cabo.

- Recuperación de imágenes basadas en contenido: conocido como *content-based image retrieval*. Consiste en buscar un conjunto de imágenes con un contenido específico de entre un conjunto de imágenes de diversa temática. El tipo de contenido se puede establecer mediante otras imágenes o texto. Un ejemplo de esto es *Google Goggles* que realiza búsqueda de imágenes a través de una similar.

2.2. Estimación de flujo óptico

La estimación del flujo óptico consiste en *"calcular una aproximación del campo de movimiento en 2D (...) a partir de patrones espacio-temporales de la intesidad de imagen"* [12]. Dado que es un tema muy presente en la literatura, esto ha llevado al desarrollo de varias técnicas para su estimación.

2.2.1. Técnicas diferenciales

Las técnicas de estimación del flujo óptico que utilizan derivadas parciales de la imagen o versiones filtradas de éstas, son las llamadas diferenciales. Dentro de este tipo de métodos, existen varios métodos como los de Buxton-Buxton [5] y Black-Jepson [4]. Sin embargo, los más populares, que detallaremos a continuación, son los de Lucas-Kanade [19] o Horn-Schunck [8].

Lucas-Kanade

Como bien se ha referido, esta técnica es diferencial. Asume que el flujo óptico situado alrededor del píxel que se analiza es constante, es por ello que la fórmula del flujo óptico se mantendrá dentro de una ventana que está centrada en el píxel en concreto. Para resolver la

ecuación utiliza un método de mínimos cuadrados con pesos. Dado que este método es local, no necesitamos conocer toda la imagen para determinar el flujo óptico fiable de una región [24]. Al contrario que en el método de Horn-Schunck, éste funcionará mejor por tanto en el caso de imágenes heterogéneas, pues sus regiones no tendrán relación con el resto de la imagen.

Horn-Schunck

A diferencia del método de Lucas-Kanade, en [8] se asume un flujo óptico suave en toda la imagen. Éste se consigue mediante la minimización de una función de energía que depende de una constante de regularización llamada α, cuyo aumento lleva a un incremento de la suavidad del flujo. Este método normalmente se resuelve mediante las ecuaciones multidimensionales de Euler-Lagrange. La ventaja de que este método sea, al contrario que el de Lucas-Kanade, iterativo sobre una asunción global de la imagen, implica que funcionará mejor en la detección del flujo óptico en imágenes homogéneas [24].

2.2.2. Correlación de fase

Los métodos de correlación de fase asumen que los valores de los píxeles desplazados no cambiarán de un frame a otro. Por ello se puede establecer algún tipo de medida que comparen las regiones cercanas de cada píxel a estimar. En el momento en que la similitud de este valor en el segundo fotograma se maximiza, se considerará que el centro de esa región es el píxel que se buscaba, y la distancia entre este píxel y el de la imagen de partida será el vector de flujo óptico.

Barnard and Thompson

Este método utiliza características para determinar el flujo, por lo que es más elaborado, pero también aplica la correlación de fase para determinar la decisión final sobre cuál será el vector. Esta estimación se mejora tras un etiquetado de los píxeles que se basa en la similitud de regiones alrededor de cada píxel.

2.2.3. Método de bloques

Existen varias técnicas de cálculo de flujo que utilizan bloques. Una de ellas es la búsqueda exhaustiva, que definiendo una región del primer fotograma, busca una correspondencia en todas las soluciones posibles del segundo, lo cual es altamente costoso y poco eficiente. También existe la técnica del gradiente descendiente, que se guía por el gradiente del error del bloque elegido con el que se compara.

2.3. Evaluación de la estética

A través de los datos obtenidos mediante las diferentes disciplinas englobadas en computer vision, se puede obtener información de aquello que los humanos perciben visualmente. Relacionando esta información objetiva con las respuestas subjetivas de los usuarios, se puede llegar a averiguar qué características de aquello percibido son las que estimulan a la persona que lo visualiza, aunque sean implícitas y el usuario realmente no repare en ellas. En relación con esto existen muchos trabajos centrados en la evaluación de la estética, tanto de imágenes como de vídeos.

A nivel de imágenes, podemos encontrar [30], un artículo centrado en la evaluación estética de fotos mediante la extracción de características de colorido (relacionado con la saturación y el brillo), contraste, simetría o número de caras, entre otros. Sin embargo, en [18] se lleva a cabo obteniendo características genéricas basadas en contenido. Esto implica no recurrir a la evaluación de técnicas fotográficas como podría ser la regla de los tercios, sino que se centra más en el contenido semántico de las imágenes observadas.

Dado que el concepto de los los vídeos en *streaming* es relativamente nuevo y debido a que es éste uno de los motivos principales que llevan al análisis de su estética, el número de trabajos desarrollados en esta materia no es tan alto como el de otras, debido al escaso tiempo que lleva siendo un campo de interés para la investigación. Sin embargo, se pueden resaltar trabajos como [3] en donde se clasifican de manera automática un conjunto de vídeos en alto o bajo nivel de apariencia estética. Para esto se evaluaron 160 vídeos mediante un estudio y se extrajeron diferentes características, seleccionando las

más discriminativas. Utilizando características propias de los vídeos, en [9] se preprocesan con una estimación del movimiento mediante el algoritmo de estimación de flujo óptico desarrollado en [16], y se extraen características relacionadas con el espacio de movimiento, el tipo de movimiento de cámara y el nivel de temblor de ésta. Existen también trabajos que diferencian entre características a nivel de plano y de frame, como puede ser [26].

2.3.1. Etiquetado de bases de datos

A la hora de escoger la base de datos con la que se va a trabajar, es importante tener en cuenta el objetivo que se busca alcanzar con ella, por ello es necesario escoger aquel grupo de, en este caso, vídeos, para lograr comprobar si el algoritmo propuesto funciona como se espera. Sin embargo, esto conduce a la desventaja de obtener resultados poco fiables a la hora de aplicar el algoritmo a un elemento que no formase parte de la base de datos, por lo que puede tener limitaciones que deberán contemplarse. Esto implica que la búsqueda de bases de datos con un fin concreto es, en sí misma, una tarea aparte. Por ello existen artículos relacionados con la búsqueda de bases de datos propicias para cada campo, como el del flujo óptico [25].

Una tarea aún más laboriosa relacionada con esto es el etiquetado de las bases de datos de los estudios relacionados con la evaluación estética de vídeos o imágenes. Esto es provocado por los gustos y preferencias subjetivos de cada persona, lo que puede llevar a que un etiquetado supervisado provoque etiquetas ambiguas. En [11] se lleva a cabo un etiquetado de una base de datos variada y amplia ayudado por observaciones humanas. Para ello se considera la anotación de objetos, sucesos y estabilidad de la cámara. Así como en [22] se provee una base de datos para el análisis visual que proporciona tres tipos de anotaciones automáticas: estéticas, semánticas y de estilo fotográfico, de nuevo mediante la ayuda de un sistema de evaluación humano.

2.3.2. Detección de movimiento de cámara

En el cine los movimientos de cámara son un recurso que aumenta el dramatismo de las escenas. Así, por ejemplo, movimientos

de cámara de izquierda a derecha resultan más naturales debido a la forma en la que leemos. Cuando nos alejamos de algo puede ser por miedo, lo cual se representa mediante un zoom out. Si queremos focalizar algo, lo simbolizaremos con el zoom in. Sin embargo, si el objetivo es transmitir una mala sensación, obtendremos planos con cierto movimiento irregular como el que puede proporcionar una cámara al hombro. Por el contrario, en un anuncio puede que convenga que la cámara sea fija dado que eso permitirá mostrar con claridad el producto que se pretende publicitar.

Como podemos ver, la relación del movimiento de cámara con la percepción estética del vídeo es muy estrecha, por lo que es interesante detectarlo como una característica más. Además, su obtención ayudaría en otras materias como la indexación de vídeos.

En [14] se detecta un movimiento global provocado por el cambio de enfoque o el movimiento de cámara. Para esto se utiliza un método afín de seis parámetros y posteriormente se calculan los vectores de compensación de movimiento. A partir de estos parámetros se obtienen seis movimientos: pan (horizontal), tilt (vertical), zoom, rotación y dos tipos de hiperbólicos, teniendo cada una de las cantidades obtenidas diferente significado: en el caso de pan y tilt se refiere al número de píxeles y el resto representan relaciones. Posteriormente se realiza una separación del vídeo en planos y segmentación del movimiento, clasificando únicamente aquellos que detectan pan, tilt y zoom.

De manera similar, en [15] se obtiene una parametrización del movimiento que consta de ocho parámetros. De la misma manera, se definen valores relacionados con estos parámetros que determinarán los diferentes movimientos de cámara. Posteriormente se debe considerar qué movimientos tienen significado en relación con la percepción humana, lo cual implica relacionarlo con la cantidad de movimiento y su duración temporal. Para esto es necesario determinar el valor de umbral en las magnitudes de cada movimiento de cámara, así como la duración de éstos, para el cual se determinará su presencia o no a nivel de plano. Se etiquetan manualmente un conjunto de vídeos de la base de datos, y se realizan pruebas para determinar estos valores.

Otros trabajos, como [20] obtienen parámetros de movimiento

de cámara mediante el flujo óptico. Esto se hace estableciendo una relación entre el flujo con cada tipo de movimiento. Se considera por ejemplo que, un movimiento en horizontal como es el pan, producirá por tanto un flujo en esa dirección, y vertical en el caso de un tilt.

2.4. Aprendizaje máquina

El aprendizaje máquina o *machine learning* es un campo de la inteligencia artificial estrechamente relacionado con la estadística computacional y cuyo objetivo es desarrollar algoritmos que sean capaces de aprender de patrones y hacer predicciones a partir de datos. Un ejemplo de esto es la detección de correo basura o *spam*, ya que no se puede detectar mediante la búsqueda de patrones ya establecidos, debido a que éstos varían con el tiempo y según su destinatario y origen. Sin embargo, mediante el aprendizaje máquina se pueden obtener estos nuevos patrones a partir de varias muestras de correos que se corresponden con *spam*, y considerar como tales aquellos correos futuros cuyos patrones sean similares.

2.4.1. Aplicaciones

En [2] se aportan diferentes ejemplos de aplicaciones del aprendizaje máquina con los que será más fácil comprender sus utilidades. De estos podemos resumir los más importantes:

- **Clasificación**: teniendo unos datos de entrada, se clasifica cada caso en un tipo u otro tras algún tipo de cálculo con ellos.
- **Predicción**: teniendo ya una clasificación realizada, si los acontecimientos que se quieren predecir son similares a los pasados, se podrá determinar la pertenencia del nuevo evento a una u otra clasificación.
- **Reconocimiento de patrones:** un ejemplo es la detección facial en imágenes. Aunque no todas son iguales, todas corresponden a ciertos patrones relacionados con la estrutura general de una cara: ojos, cejas, mandíbula, etc. Este aprendizaje se puede aplicar a todo tipo de datos, así como detección de letras o diagnóstico médico.

- **Regresión:** cuando se tiene una serie de datos, se determina una función que se ajusta a su distribución para poder predecir el comportamiento de otros futuros valores debido a que se asume que se distribuirá de manera similar a la función de ajuste.
- **Agrupación o *clustering***: es un tipo de estimación de densidad en el que se hacen diferentes grupos de datos en función a su perfil, como puede ser la agrupación de usuarios mediante un rango de edad.

2.4.2. Algoritmos de aprendizaje supervisado

Cuando se utiliza el aprendizaje máquina, se pueden encontrar dos situaciones: tener una base de datos no etiquetada, lo que llevará a un aprendizaje no supervisado o *un-supervised learning*, o que ésta sí esté etiquetada, lo que proporcionará etiquetas a cada una de las instancias a estudiar.

Debido a la amplia variedad de algoritmos proporcionados en aprendizaje máquina, en este caso nos centraremos en conocer algunos de los algoritmos de aprendizaje supervisado que más adelante serán los utilizados en la realización del trabajo.

Naive Bayes

Dado a que es poco práctico aprender clasificadores Bayesianos debido al gran número de parámetros que se necesitan calcular, según [21] el algoritmo de Naive Bayes los reduce mediante una asunción de independencia condicional que disminuye de manera drástica los parámetros necesarios a la hora de modelar.

Regresión logarítmica

La regresión logarítmica o *logistic regression* es un algoritmo de aprendizaje máquina para funciones condicionadas del tipo $P(Y|X)$ para el caso en el que Y es discreta, y X un vector con valores continuos o discretos, en cual se estiman probabilidades con una función logarítmica. En el caso de un etiquetado binario, este algoritmo proporciona una regla de clasificación lineal [21].

Classification And Regression Trees (CART)

Los algoritmos de árboles de regresión y clasificación representan los datos mediante divisiones cada vez más pequeñas con preguntas binarias. Los algoritmos *CART* harán una búsqueda a lo largo de todos los valores para encontrar la división más homogénea. Esto se hace de manera iterativa en cada fragmento de datos obtenido, según se explica en [29].

Sequential Minimal Optimization (SMO)

Como se explica en [13], la complejidad del modelado de un *Support Vector Machine (SVM)* no se ve afetada por el número de atributos elegidos en el conjunto de entrenamiento, por lo que es una técnica adecuada a la hora de trabajar con un gran número de atributos. Sin embargo, este método se lleva a cabo resolviendo un problema de programación cuadrática o *quadratic programming (QP)* de N dimensiones, siendo N el número de muestras evaluadas, lo cual implica operar con grandes matrices conllevando un gran gasto de tiempo de computación. Como solución a esto, aparece el algoritmo *SMO* que resuelve el problema de manera relativamente rápida sin necesidad de almacenar grandes matrices ni utilizar optimizaciones numéricas. Esto se consigue descomponiendo el principal problema *QP* en varios sub-problemas.

3. Base de datos

Dado que este trabajo está relacionado con el hecho en [7], se utiliza la misma base de datos. En las siguientes secciones se explicará brevemente el proceso llevado a cabo para su selección y etiquetado.

3.1. Dominio

Para que la información obtenida de los vídeos no esté muy sesgada, es necesario que tengan características similares entre sí, relacionadas tanto con el contenido semántico de los vídeos como con sus metadatos y duración.

Se eligen en concreto vídeos de anuncios de coches, debido a que el público que los ha visionado tendrá en general el mismo tipo de interés y el objetivo de los vídeos es común. Además el contenido de los anuncios en general es poco variado. Esto hace que su éxito resida en gran parte en la manera en la que se presente el producto y por tanto, en sus características estéticas, aunque sin ser posible ignorar el sesgo proporcionado por las preferencias de cada espectador.

3.2. Filtrado

Se escoge *YouTube* debido a la variedad de metadatos y vídeos que puede proveer para formar la base de datos y su etiquetado.

Como comienzo se buscan únicamente anuncios de marcas vendidas en España que estén español, así como vídeos a partir del año 2010 para evitar sesgo cultural y temporal que pueda afectar a los datos obtenidos. Así, mediante búsquedas con palabras clave, se obtienen 2732 vídeos que necesitan ser filtrados, por lo que se eliminan aquellos repetidos así como los que no pertenecen a la categoría seleccionada, limitando la duración del vídeo a aquellos entre 10 y 115

segundos. Además se descartan los que carecen de los metadatos necesarios para obtener una etiqueta adecuada. Tras esto se obtienen 277 vídeos que son filtrados de manera manual para comprobar el cumplimiento de los requisitos, eliminando aquellos con poca calidad, virales o que no se ciñen al dominio seleccionado.

Este filtrado lleva a la obtención de un conjunto de 138 vídeos que carecen de etiquetado, por lo que el siguiente paso será establecer un sistema de etiquetas para cada uno de los vídeos, el cual se basará en los metadatos proporcionados por *YouTube*.

3.3. Anotación

A partir de los metadatos proporcionados por la plataforma se obtienen diferentes datos para cada vídeo, como se puede ver en la tabla 3.1.

viewsCount	Veces que se ha visualizado el vídeo
numLikes	Veces que los usuarios han indicado que les gusta el vídeo
numDislikes	Veces que los usuarios han indicado que no les gusta el vídeo
favoriteCount	Veces que los usuarios han añadido el vídeo a favoritos
rating	Evaluación media de 1 a 5 estrellas en saltos de 0.5
numRaters	Usuarios que han evaluado con estrellas o *like/dislike*
numComments	Número de comentarios del vídeo
IdRatio	Número de *likes* respecto de *likes+dislikes*
viewsCountScore	Evaluación media de 1 a 5 según el número de visitas

Tabla 3.1: Datos utilizados para el etiquetado

Estos datos se pueden clasificar en dos tipos: implícitos, que son los proporcionados por el usuario por el simple hecho de haber visitado el vídeo, al que sólo pertenecerán *viewsCount* y *viewsCountScore*; y explícitos, al que pertenecerán el resto y del cual el usuario es consciente a la hora de proporcionar la información. A partir de estos datos, se obtendrán tres tipos de etiquetado.

- **Calidad**: a partir del dato explícito *IdRatio*, se calcula la mediana. Eso es obtener el valor para el cual la mitad de los vídeos queda por encima de éste y la mitad por debajo. En este caso, aquellos vídeos superiores al valor de la mediana, serán considerados con una estética positiva.

- **Cantidad**: de manera similar, se calcula la mediana del dato implícito *viewsCount*. Los vídeos cuyo valor esté por encima de la mediana, serán considerados como vídeos estéticamente positivos o con impacto.
- **Combinación**: se aplican las dos etiquetas previas de manera conjunta, obteniendo un etiquetado de 4 clases, una por cada combinación según el grupo al que pertenezca en cada uno de los dos métodos previos.

4. Estudio del flujo óptico

El flujo óptico, como se explicó en el apartado 2.1.2, alude al aparente patrón de movimiento de una escena, ya sea de los objetos presentes en ésta o de sus bordes, que es provocado por el movimiento respecto del ojo o de la cámara que lo captura.

La estimación automática del flujo óptico es una tarea complicada, pero existen varios métodos para llevarla a cabo. Sin embargo, debido a su disponibilidad de código en *Matlab* y su eficacia, el elegido será el algoritmo utilizado en la tesis doctoral [16].

Para obtener una estimación de flujo óptico óptimo para nuestra base de datos, la cual difiere de la utilizada para el desarrollo de este algoritmo, es necesario además elegir correctamente sus parámetros no definidos previamente y filtrar o eliminar aquellos píxeles que podrían estar estimados erróneamente debido a su baja presencia de textura (ver figura 4.1).

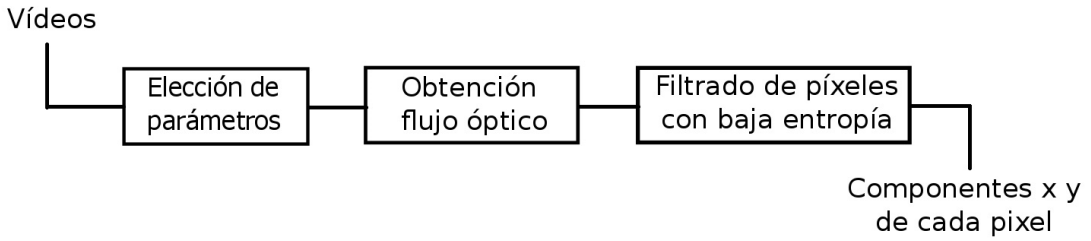


Figura 4.1: Pasos para la obtención del flujo óptico

4.1. Obtención del flujo óptico

En [16] se busca, entre otras cosas, demostrar que para el análisis del movimiento es más útil buscar la correspondencia de estructuras

locales entre imágenes, antes que de intensidad entre píxeles como se ha hecho tradicionalmente en esta materia.

Para su desarrollo, se estableció una base de datos consistente en vídeos con movimientos reales y se propuso una solución a la limitación de vídeos en espacio cerrado y con movimiento irreal propuesta por *Baker et al* [25]. Esto se hace mediante un sistema de anotación del movimiento llevado a cabo por humanos, que proveyó a las secuencias de vídeo de un etiquetado o *ground-truth*. Para esto es útil conocer el flujo óptico entre dos fotogramas o *frames*.

4.1.1. Funcionamiento del algoritmo

Para simplificar la formulación del problema del cálculo del flujo óptico, tradicionalmente es común asumir la constancia de brillo de un mismo píxel entre dos frames consecutivos. Gracias a esto se obtiene una función objetivo (ver ecuación 4.1) dependiente del gradiente del campo de flujo o *flow field*, y de α que aporta un peso a la regularización y suaviza los vectores. Debido a que se obtiene una función no convexa, para evitar que la opmitización lleve a un mínimo local se lleva a cabo el procedimiento *coarse-to-fine* (ver 4.1.2) con una pirámide Gaussiana deformada -es decir, con *warping*- cuya resolución se puede disminuir (*downsample*).

$$E(u,v) = \int \psi(|I(p+w) - I(p)|^2) + \alpha\phi(|\nabla u|^2 + |\nabla v|^2)dp \quad (4.1)$$

La ecuación 4.1 es complicada de optimizar ya que es una función continua que depende del espacio y del tiempo. Por esto se discretiza y se linealiza mediante la expansión de Taylor. Posteriormente se elimina el factor temporal ya que sólo se pretende calcular el flujo óptico de dos frames. Finalmente se vectorizan los componentes de la función obteniendo la ecuación 4.2.

$$\begin{aligned} E(dU,dV) = \sum_P \psi\left((\delta_P^T(I_z + I_x dU + I_y dV))^2\right) + \\ \alpha\phi((\delta_P^T D_x(U+dU))^2 + (\delta_P^T D_y(U+dU))^2 + \\ (\delta_P^T D_x(V+dV))^2 + (\delta_P^T D_y(V+dV))^2) \end{aligned} \quad (4.2)$$

Se utiliza el algoritmo *Iterative Recursive Least Squares (IRLS)* para encontrar aquellos valores que hacen que se cumpla que el gradiente respecto de las componentes del flujo sea nulo (véase la ecuación 4.3). Con este método se recalculan los pesos de las variables ψ y ϕ para posteriormente utilizar una variante del método de Gauss con una convergencia más rápida y resolver el sistema lineal resultante: el *Successive Over-Relaxation (SOR)*.

$$\left[\frac{\delta E}{\delta dU}; \frac{\delta E}{\delta dV}\right] = 0 \tag{4.3}$$

En este trabajo se aplica un código de *C++* utilizable en *Matlab* que implementa una función *Coarse2FineTwoFrames*, cuya finalidad es obtener las componentes x e y del vector de movimiento de los píxeles de cada imagen. Este código ha sido publicado con propósitos educacionales y de investigación, por lo que se puede aplicar en este trabajo. De lo contrario, sería necesario buscar una alternativa similar o implementar una de las técnicas explicadas previamente en el apartado 2.2.

4.1.2. Método coarse-to-fine

Para evitar llegar a un mínimo local en la optimización, se aplica el método *coarse-to-fine*. Este consiste en calcular aproximadamente el flujo de una versión de la imagen con menor tamaño y por ello menor definición, a lo que se le llamará *coarse level*, para posteriormente propagarlo a un nivel con mayor resolución o *fine level*.

En el nivel más alto de la pirámide, el de menor tamaño (*coarsest level*), se establece una ventana cuadrada y la coordenada del píxel que se quiere buscar. Una vez se calcula el flujo, se propaga el vector al siguiente nivel, que tendrá más resolución (*finer level*). Así se continua con este procedimiento hasta llegar a la imagen con el tamaño original, obteniendo el flujo óptico con menor coste computacional, dado que su estimación se ha llevado a cabo en una imagen de menor tamaño.

El gran inconveniente de este sistema es que los vectores estimados en el tamaño final de los fotogramas son tan fiables como aquellos calculados en el primer nivel. Si estos contienen muchos fallos en la

estimación, se propagarán en cada operación, acumulándose en el nivel final [23].

4.2. Parámetros

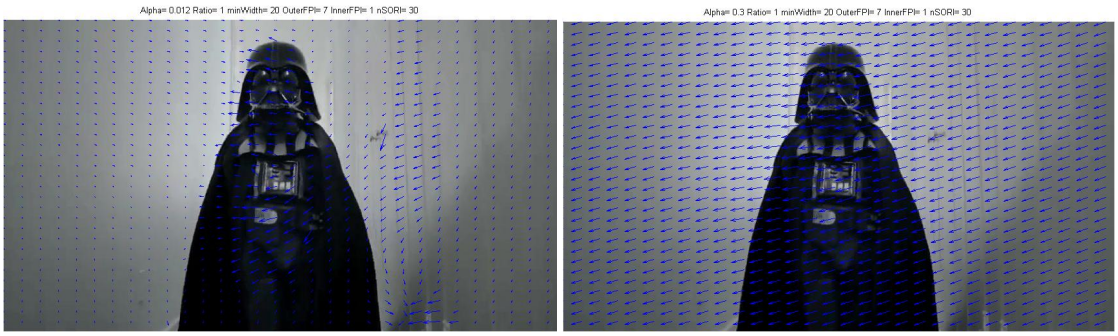
Tras conocer el funcionamiento del algoritmo utilizado, se puede entender fácilmente la utilidad de los parámetros variables de la función *Coarse2FineTwoFrames*. Para poder utilizarla es necesaria la fijación de seis parámetros que influirán de manera notable tanto en el resultado obtenido como en el tiempo empleado en el cálculo de éste.

Debido a la complejidad del problema, esta tarea se ha de llevar a cabo de manera manual por la autora, ya que es necesario establecer una correspondencia entre el flujo óptico estimado y el visualizado.

- **Alpha (α)**: tiene su origen en el método diferencial de Horn-Schunck referido en el apartado 2.2.1, en donde actúa como término de regularización. Cuanto más crezca el valor de α, mayor suavidad tendrá el flujo, debido a que se penalizan en mayor cantidad los gradientes más altos, haciéndolos menos pronunciados y suavizando así el flujo representado. Dado que este término es constante, en aquellas imágenes en las que predominan grandes gradientes, α hace que éstos cobren menos importancia. Esto es debido a que el valor calculado distará más de su valor "real" que en los casos de menor gradiente -que serán más suaves-, por lo que este parámetro actuará en cierto modo de peso, dándole así uno mayor a aquellas regiones con un gradiente más bajo [6].

  En 4.2 podemos observar como en la figura 4.2a, al tener un valor de α más bajo, los vectores de flujo óptico son más heterogéneos que en el caso de la figura 4.2b, que representa un campo óptico mucho más suave pero a su vez, poco representativo del movimiento.

- **Ratio:** se refiere a la *downsampling ratio* o la relación de disminución de la resolución en la imagen referida en la pirámide Gaussiana. Esto implica que un valor de 0.5 disminuirá la calidad de la imagen a la mitad, 0.25 a un cuarto o fijarlo en 1 haría



(a) Flujo óptico con α=0.012 (b) Flujo óptico con α=0.3

Figura 4.2: Efectos de α en el flujo óptico

que no se perdiera información y se mantuviese la resolución. El hecho de que este parámetro sea menor a 1 implicará perder datos sobre los píxeles de la imagen original, algo que puede suponer errores a la hora de la estimación del flujo óptico, pero a su vez un menor coste computacional comparado al de la imagen original.

- **minWidth:** este parámetro indica la anchura del *coarsest level* ya comentado en el apartado 4.1.2.
- **nInnerFPIterations** y **nOuterFPIterations:** dado que en el algoritmo de optimación del flujo óptico utilizado se aplica el método *IRLS*, se utiliza una equivalencia propuesta en [28]. Aquí se determina que esta técnica equivale a la aplicación de dos iteraciones para el cálculo de los valores de ψ y ϕ, los cuales vienen determinados por estos parámetros y que corregirán la carencia de linearidad, primero de los símbolos I (iteración externa u *outer*) y después la de ψ (iteración interna o *inner*).
- **nSORIterations:** tras la aplicación de las dos iteraciones previas, se obtiene un sistema de ecuaciones para el cual se utiliza el método *SOR*, una variación de la resolución de *Gauss-Seidel* con una convergencia más rápida.

Tras conocer la función y aplicación de cada uno de los parámetros utilizados, es necesario encontrar un valor para cada uno de ellos que se adapte de la manera más óptima posible a los vídeos de nuestra base de datos. Para esto se ha llevado a cabo un visionado

exhaustivo de diferentes combinaciones de fotogramas consecutivos pertenecientes o no al mismo vídeo. Se parten de los parámetros por defecto proporcionados en [16]:

α	0.012
Ratio	0.75
minWidth	20
nInnerFPIterations	1
nOuterFPIterations	1
nSORIterations	30

Tabla 4.1: Valores de partida para los parámetros

4.2.1. Alpha (α)

Para el visionado de los vectores mediante flechas, se utilizan las componentes de x e y obtenidas a partir método *Coarse2FineTwoFrames* y la función *quiver* que proporciona *Matlab*. A ésta se le introducen como parámetros las dos matrices de las componentes del vector de movimiento de cada píxel y dos vectores de puntos que darán las coordenadas en las que se representarán cada uno de ellos. Esto es únicamente para hacer posible la visualización, ya que de no hacerlo, se representaría cada vector en cada uno de los píxeles y no sería diferenciable.

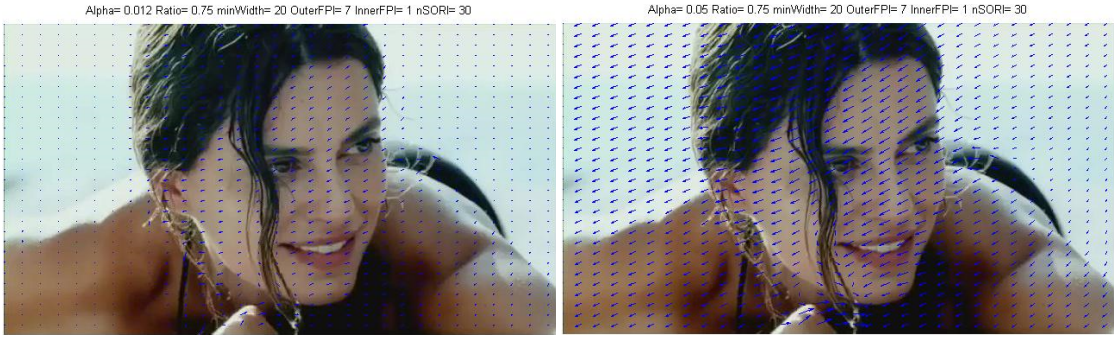
Una vez realizada la visualización del flujo óptico y partiendo del valor asignado para los parámetros, probamos el valor ya asignado por defecto (figura 4.3a). Tras esto, variamos únicamente α para ver cómo afecta. Primero elegimos un valor mucho mayor que el de partida como en 4.3b. Dado que en 4.3 observa una suavización excesiva del flujo, hasta tal punto que no es representativo, decidimos decrementar el valor del parámetro drásticamente en 4.4b, pero aún siendo mayor que el de partida.

De nuevo, a pesar del decremento del valor, éste sigue siendo demasiado alto. Probamos un valor drásticamente bajo para ver su funcionamiento, como se ve en la figura 4.5b.

En el caso de la comparación en la figura 4.5, el decremento ha hecho aún menos visibles los vectores del flujo óptico, por lo que deja igualmente de ser significativo. Tras aplicar estas variaciones a

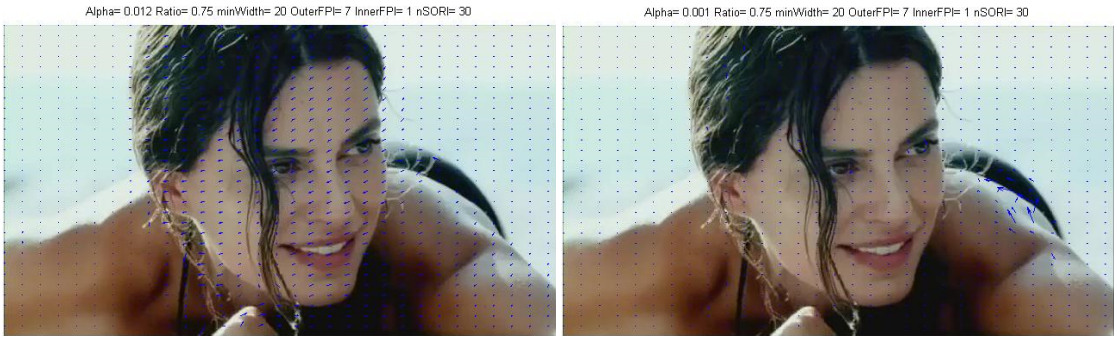
(a) Flujo óptico con α=0.012 (b) Flujo óptico con α=0.3

Figura 4.3: Comparación del incremento de α a 0.3



(a) Flujo óptico con α=0.012 (b) Flujo óptico con α=0.05

Figura 4.4: Comparación del incremento de α a 0.05



(a) Flujo óptico con α=0.012 (b) Flujo óptico con α=0.001

Figura 4.5: Comparación del decremento de α a 0.001

varios conjuntos de frames se determina fijar el valor de α en 0.012, aquél aplicado por defecto, ya que parece aplicar una suavidad lo suficientemente apropiada como para que los vectores sean claros y representativos.

4.2.2. Ratio

Como bien se comentó en 4.2, un valor muy pequeño de este parámetro puede llevar a un cálculo erróneo del flujo óptico. De nuevo partimos de los parámetros prefijados y lo variamos para visualizar cómo afecta.

Dado que previamente podemos aventurar que el resultado óptimo se producirá cuando el valor del ratio sea 1, se observa también el tiempo invertido en finalizar la tarea, siempre bajo las mismas condiciones, para ayudar a la decisión.

(a) Flujo óptico con ratio = 0.75 (b) Flujo óptico con ratio = 1

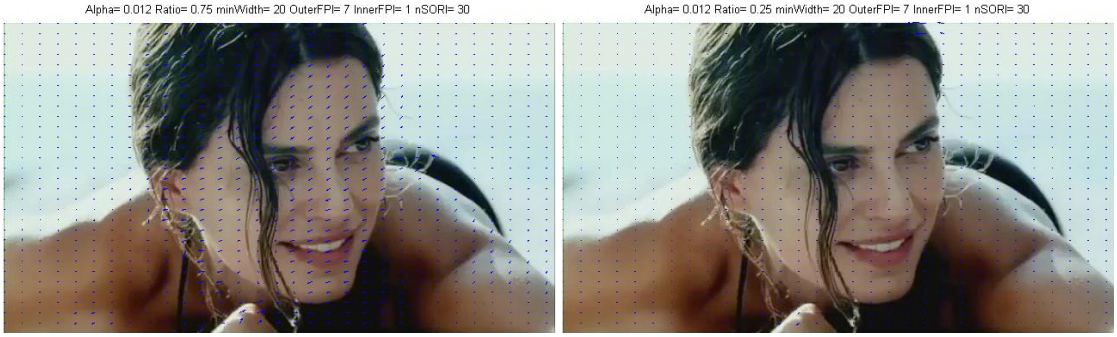
Figura 4.6: Comparación del incremento del ratio a 1

Tanto la figura 4.6a como la figura 4.6b son similares, aunque podría parecer que ésta última tiene unos vectores más "definidos". Se calcula el tiempo transcurrido en el cálculo de la función: en el primer caso se obtiene 7.80 segundos y en el segundo 7.85 por lo que no hay una diferencia considerable entre ambos. Esto es variable según el tamaño de los frames y empieza a ser un dato más relevante si se tienen en cuenta todos los fotogramas que forman parte del vídeo, dado que el tiempo de cálculo entre un valor y otro sí pasaría a ser considerable en el total.

Tras esto, se prueban y se tienen en cuenta los tiempos de la reducción del ratio a 0.5 (figura 4.7) y 0.25 (figura 4.8).

(a) Flujo óptico con ratio = 0.75 (b) Flujo óptico con ratio = 0.5

Figura 4.7: Comparación del decremento del ratio a 0.5



(a) Flujo óptico ratio=0.75 (b) Flujo óptico ratio=0.25

Figura 4.8: Comparación del decremento del ratio a 0.25

En la figura 4.7b, como en la figura 4.6b, se aprecian diferencias respecto al valor por defecto, aunque no son realmente significativas. Sin embargo, la disminución de la resolución en la figura 4.8b afecta de manera significativa, haciendo apenas visibles los vectores de flujo óptico. Así, observamos el tiempo de cálculo en el primer caso y éste resulta notablemente inferior: 4.13 segundos en ambos.

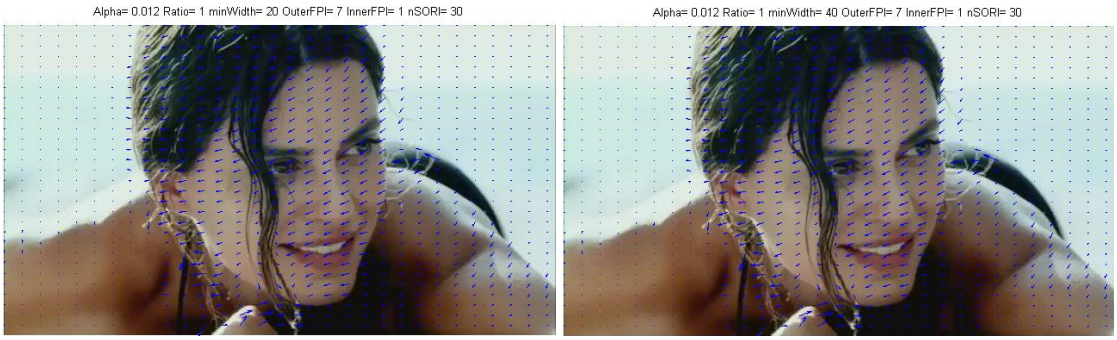
A pesar de la diferencia de tiempo computacional entre los valores de este parámetro 1 y 0.75 y el valor de 0.5, se decide fijar este parámetro a 1, dado que por definición siempre dará una solución más fiable al tener todas las muestras de la imagen intactas, a pesar del coste computacional.

4.2.3. minWidth

En este parámetro se ha observado que, dependiendo del tamaño de las imágenes que se utilizan y de lo grandes que éstas sean, no

es posible proceder al cálculo ya que *Matlab* encuentra un fallo que urge a su cierre. Se establece la relación de que esto puede ser debido a que si el valor de la anchura del nivel con menos resolución, el *coarsest level*, se aproxima lo suficiente a la anchura de los frames, no se puede realizar el cálculo de los vectores. Además de tener en cuenta esto, este parámetro da también una situación de compromiso: la anchura debe ser lo suficientemente grande como para que no se pierda demasiada resolución respecto a la imagen original, pero también lo suficientemente pequeña como para poder llevar a cabo una optimización correcta y reducir de manera notable el tiempo de cálculo de la estimación del flujo óptico en el primer nivel.

Los frames utilizados para estos ejemplos son del tamaño 640x360, algo que varía dependiendo del vídeo. Teniendo esto en cuenta, partimos del valor inicial de este parámetro y lo doblamos a 40.



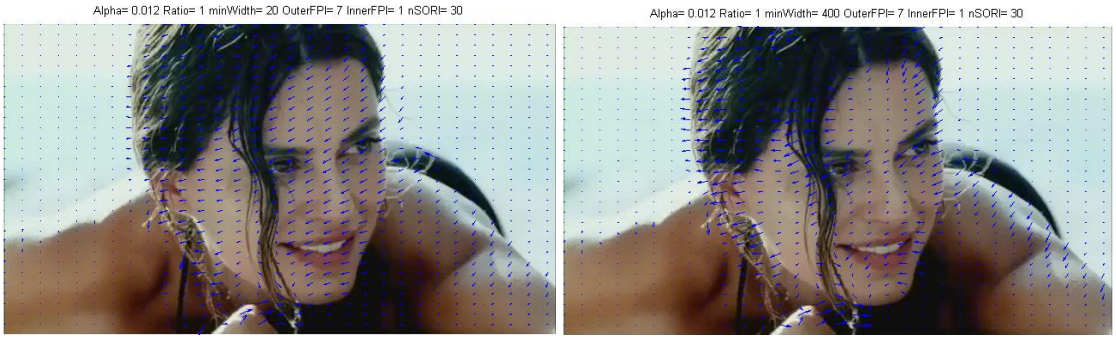
(a) Flujo óptico width=20 (b) Flujo óptico width=40

Figura 4.9: Comparación del incremento de la width a 40

Como se observa en 4.9b, la diferencia entre ambos casos es inapreciable a la vista. Tras esto, se decide aumentar más aún el valor, a 400 en este caso.

En la figura 4.10b se aprecia una diferencia con respecto a la figura 4.10a que parece definir mejor el flujo óptico. Sin embargo, tras probar en un vídeo de dimensiones 480x271, se obtiene el error previamente aludido (figura 4.11):

No es un caso excepcional. En el ejemplo previo, si se fija un valor de minWidth de 600, esto sigue sucediendo. Sin embargo, no es un problema ya que cuanto mayor sea imagen del *coarsest level*, más tiempo de procesado se necesitará y no por ello obtendremos un resultado notablemente mejor. Por ello, en lugar de establecer un



(a) Flujo óptico width=20 (b) Flujo óptico width=400

Figura 4.10: Comparación del incremento de la width a 400

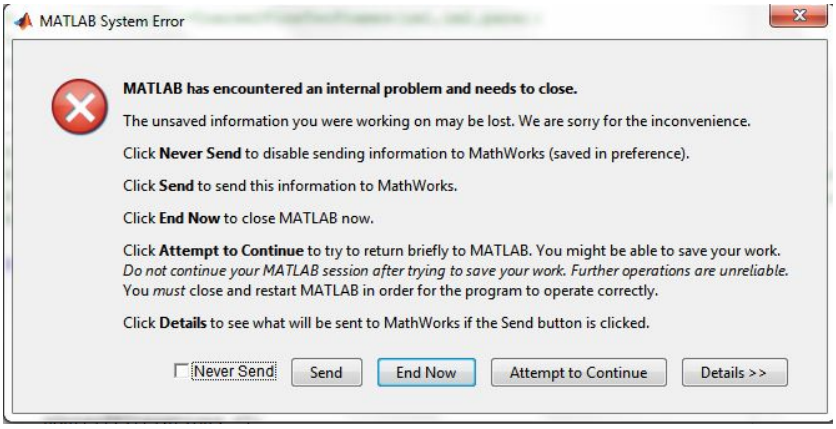


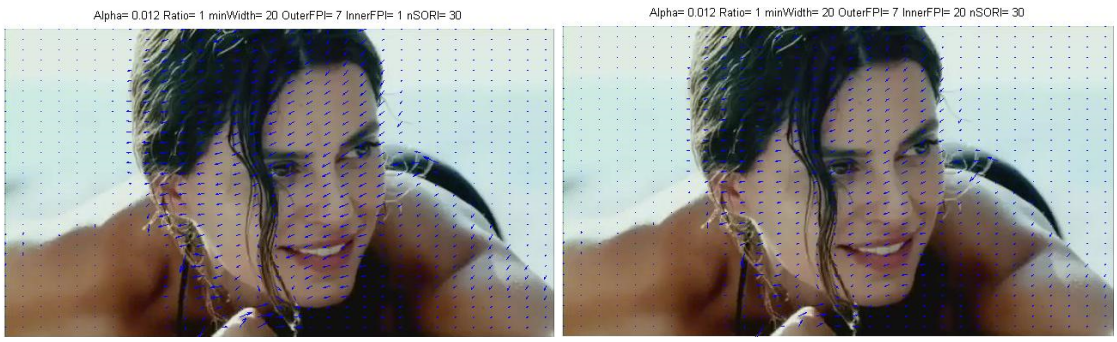
Figura 4.11: Error obtenido al usar un valor de width demasiado alto

parámetro dinámico, ya que no necesitamos obligatoriamente valores tan grandes, se decide fijar el valor predeterminado de 20, pues es apto para todos los tamaños.

4.2.4. nInnerFPIterations y nOuterFPIterations

Dado que estos parámetros están relacionados con iteraciones, es importante tener en cuenta el tiempo que se tarda en realizar la estimación del flujo óptico. Empezamos elevando de manera notable el valor de nInnerFPIterations hasta 20 y comparándolo con el valor 1.

Como se puede observar la variación entre las figuras 4.12a y 4.12b no es apreciable, y sin embargo mientras que en el primer caso el tiempo requerido para finalizar la operación es 7.84 segundos, en el segundo aumenta drásticamente hasta 100. Esto implica que el gasto de tiempo y computación no compensa con el resultado obtenido, por

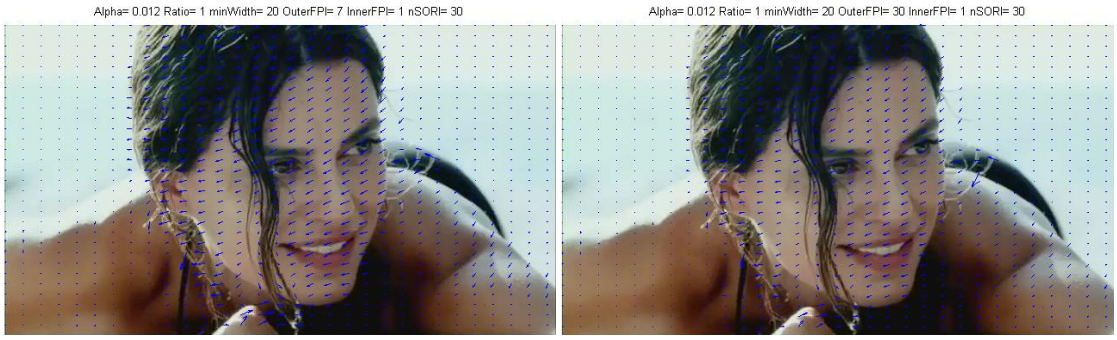


(a) Flujo óptico nInnerFPIterations=1 (b) Flujo óptico nInnerFPIterations=20

Figura 4.12: Comparación del incremento nInnerFPIterations a 20

lo que se fija nInnerFPIterations al valor por defecto: el número 1.

Ahora variamos el valor de nOuterFPIterations hasta 30 (figura 4.13b) y 1 (figura 4.14b) y medimos sus tiempos de ejecución. Las diferencias entre la primera variación y la segunda son sutiles. Además obtenemos tiempos de 7.87 y 2.73 segundos, respectivamente. A pesar de la gran diferencia a la hora llevar a cabo el cálculo, se escoge como valor final de nOuterFPIterations el valor 7, para adquirir un compromiso de tiempo con respecto a la calidad del cálculo.



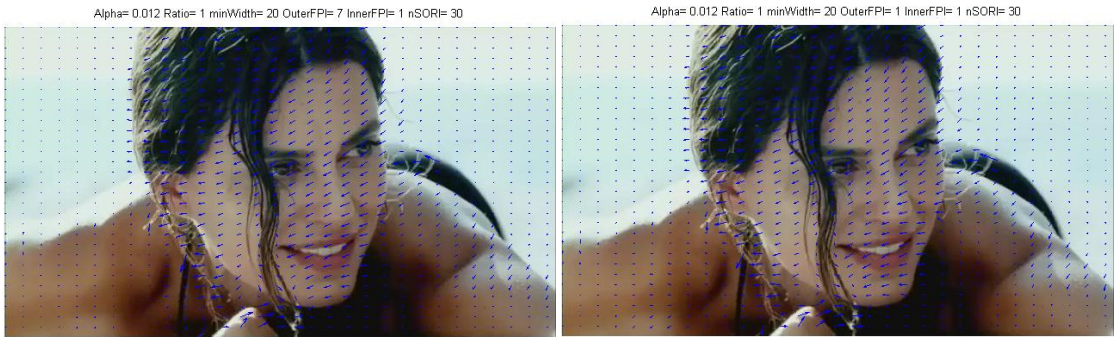
(a) Flujo óptico nOuterFPIterations=7 (b) Flujo óptico nOuterFPIterations=30

Figura 4.13: Comparación del incremento nOuterFPIterations a 30

4.2.5. nSORIterations

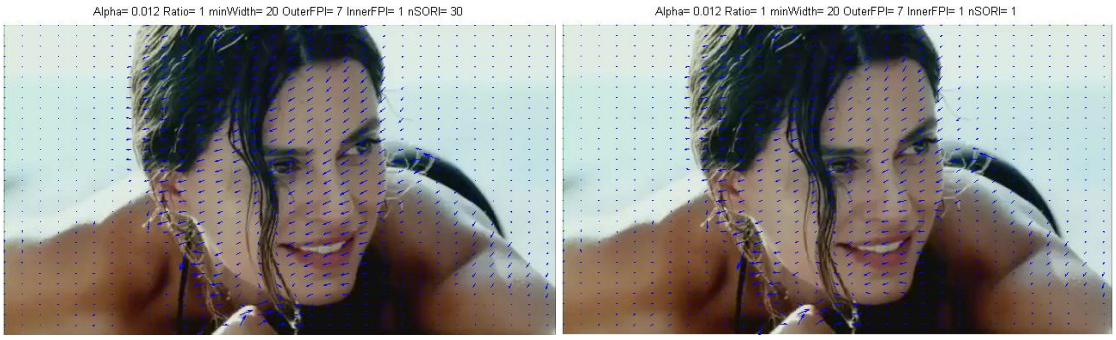
Para la fijación de este término de nuevo, al estar relacionado con las iteraciones, se tendrán en cuenta los tiempos de ejecución a la par que sus resultados visuales.

Las diferencias apreciables en las figuras 4.15 y 4.16 son ambas pequeñas. Los tiempos para los valores del parámetro 1, 30 y 50 son



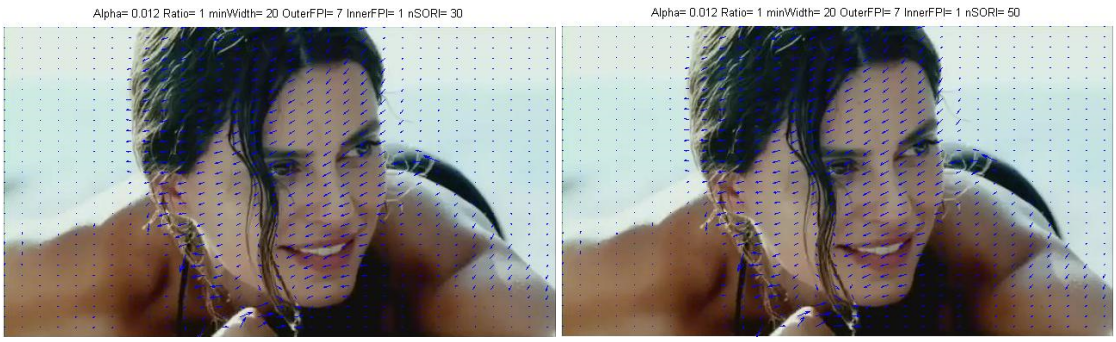
(a) Flujo óptico nOuterFPIterations=7 (b) Flujo óptico nOuterFPIterations=1

Figura 4.14: Comparación del decremento nOuterFPIterations a 1



(a) Flujo óptico nSORIterations=30 (b) Flujo óptico nSORIterations=1

Figura 4.15: Comparación del decremento nSORIterations a 1



(a) Flujo óptico nSORIterations=30 (b) Flujo óptico nSORIterations=50

Figura 4.16: Comparación del incremento nSORIterations a 50

4.85, 7.77 y 9.97 segundos respectivamente. Obviamente, el tiempo de procesado aumenta con el número de iteraciones seleccionadas y, una vez más, decidimos un valor intermedio de compromiso entre tiempo y resultados: en este caso fijaremos nSORIterations en 30.

4.2.6. Valores finales

Tras el análisis de la influencia de los parámetros a la hora de determinar el flujo óptico, se determina que los valores utilizados para la totalidad de los cálculos restantes serán los fijados en la tabla 4.2.

Parámetro	Valor por defecto	Valor final
α	0.012	0.012
Ratio	0.75	1
minWidth	20	20
nInnerFPIterations	1	1
nOuterFPIterations	1	7
nSORIterations	30	30

Tabla 4.2: Parámetros fijados para los cálculos

Todos los valores fijados son iguales a los de partida, exceptuando *nOuterFPIterations* y *Ratio*. El primero se modificó debido a un compromiso entre tiempo de cálculo y resultado, ya que un número mayor no tenía un comportamiento lo suficientemente bueno en relación a su coste computacional, y uno menor no aportaba demasiada diferencia, pero por teoría podría llegar a dar un peor resultado -como similarmente sucedió en el caso de *nSORIterations*-. En el caso del segundo parámetro, se decidió fijar a 1 debido a su relación con la disminución de resolución de la imagen que implica pérdida de información. Fijando este valor, nos aseguramos de obtener toda la información posible para la estimación del flujo óptico. En el caso de α el mejor resultado se obtuvo con el valor 0.012 debido a que aportaba una representación del flujo lo suficientemente representativa. Al tratar con *minWidth* se obtuvieron fallos cuando este era demasiado grande y por teoría fijar este valor a 20 era razonable.

4.3. Detección de regiones con baja textura

Tras la obtención del flujo óptico, el siguiente paso es eliminar aquellos píxeles cuya estimación pudiera ser errónea debido a su naturaleza de escasa textura. Como herramienta de medida para la evaluación de regiones muy homogéneas utilizaremos la entropía, que nos dará información sobre el nivel de aleatoriedad de los píxeles. Si

la entropía es baja, implicará que la región evaluada es altamente previsible, es decir, homogénea. Por lo contrario, si es alta, implicará que sus píxeles tienen un comportamiento imprevisible y estaríamos ante una región heterogénea.

La detección de estas regiones es necesaria dado que la estimación del flujo óptico del algoritmo explicado en [16] se basa en la constancia de los valores de los píxeles de un frame a otro, y puede suponer un problema en aquellas regiones homogéneas de la imagen. Esto es debido a que, si nuestro objetivo es buscar un píxel de similar valor en el próximo fotograma, al tener muchos similares alrededor, se puede establecer una correspondencia de manera errónea con otro píxel, el cual no tendrá correspondencia con el que estamos buscando.

Para proceder a la eliminación de las componentes vectoriales en regiones homogéneas, primero se ha de determinar un umbral de entropía por el cual consideremos si una región es homogénea o no. El método de fijación del umbral será nuevamente manual, ya que se necesitan obtener resultados que no eliminen demasiados píxeles, ya que de ser así, nuestros descriptores no serían representativos de los vídeos. Sin embargo, también se necesita valorar que este resultado sea coherente con las imágenes que se tratan, pues algunas necesariamente tendrán un gran porcentaje de valores eliminados debido a su naturaleza.

4.3.1. Selección del umbral

Para determinar el umbral de la entropía en cada región, el primer paso es convertir la imagen a escala de grises y dividirla en regiones. En este caso haremos 60 (ver figura 4.17) dado que se comprueba que si el tamaño es demasiado grande (pocas ventanas), las ventanas podrán englobar con más probabilidad regiones que puedan contener zonas con alta y baja textura simultáneamente, llegando a verse afectado el cálculo total de la entropía. Esto podría llevar a determinar como zona de baja entropía a esta región, cuando realmente existe alta entropía en parte, o viceversa. Además, con un alto número de regiones (ventanas más pequeñas), se prolonga el proceso de filtrado de zonas de baja textura, complicando también

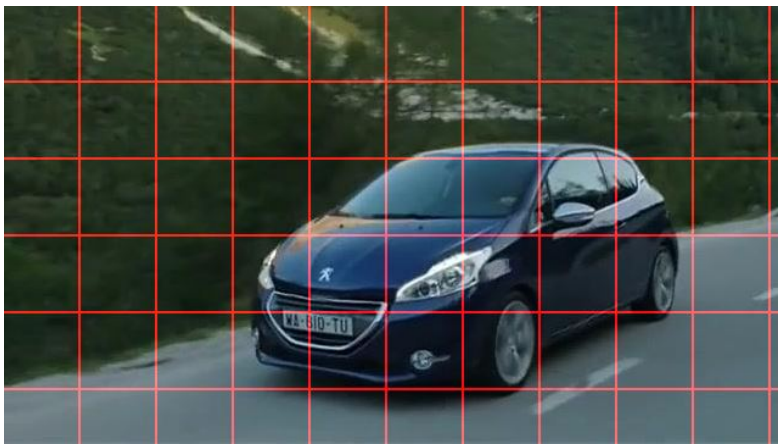


Figura 4.17: Ejemplo de la división en regiones

la visualización para poder determinar el umbral más apropiado.

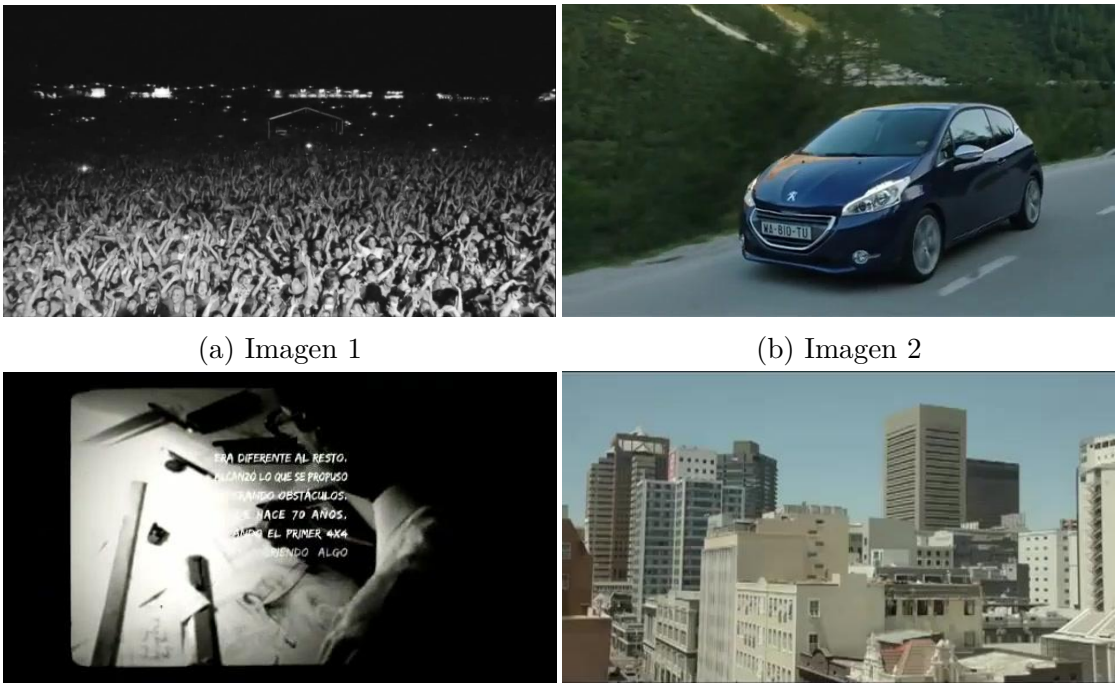
Posteriormente, se calcula la entropía de cada región, y se compara con un umbral que iremos variando para probar su eficacia. Por razones de espacio, sólo representaremos aquí cuatro de las imágenes utilizadas para esta prueba. Se escogen con regiones homogéneas (figuras 4.18a y 4.18c) y algunas claramente heterogéneas (figuras 4.18b y 4.18d) en cuanto a intensidad, aunque en 4.18d se puede observar que existe una región muy homogénea en textura: la que corresponde al cielo, así como en la imagen 4.18a se pueden ver otras zonas más heterogéneas, como la zona de luces en el tercio superior de ésta. Esto será útil para comprobar en mejor manera el funcionamiento de los umbrales.

A continuación se le aplican varios umbrales de entropía, entre ellos los valores 4, 2.5 y 1 (figuras 4.19, 4.20 y 4.21 respectivamente). Los cuadrados rojos indican las regiones de píxeles que se eliminarían de aplicarlo. En la tabla 4.3 se indican los porcentajes de píxeles eliminados en cada caso.

Imagen/Umbral	4	2.5	1
1	36.63 %	19.17 %	12.35 %
2	13.63 %	1.5 %	0 %
3	46 %	42.6 %	41.42 %
4	17.62 %	0 %	0 %

Tabla 4.3: Porcentajes de píxeles eliminados por umbral

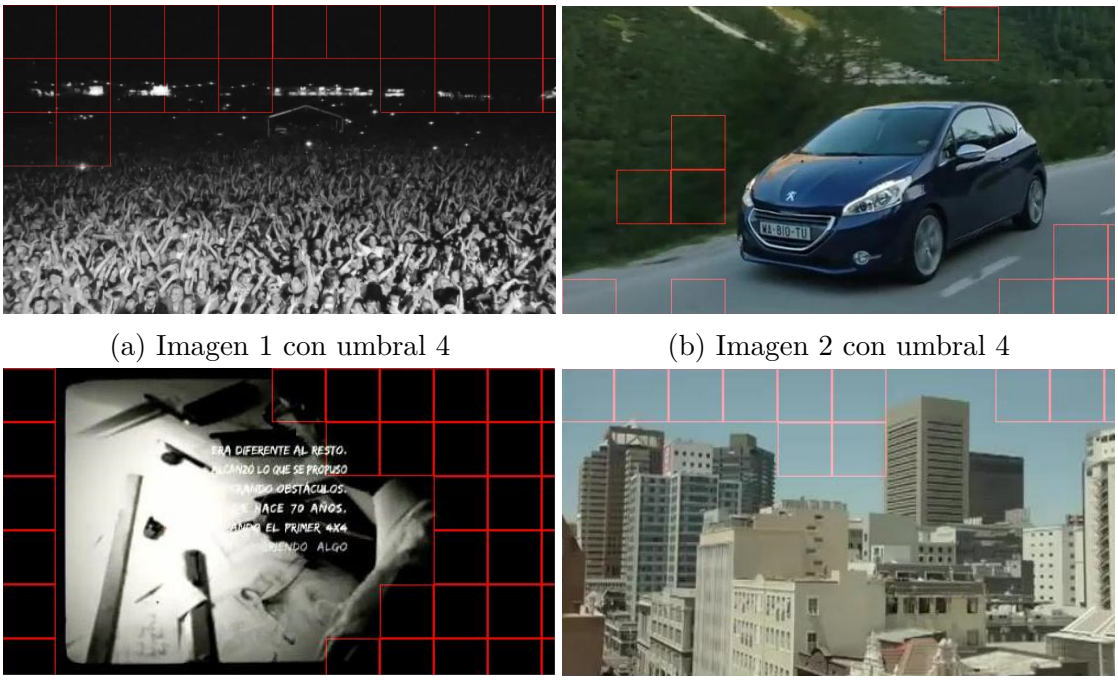
Como es obvio, desde el principio el menor porcentaje de píxeles



(a) Imagen 1 (b) Imagen 2

(c) Imagen 3 (d) Imagen 4

Figura 4.18: Imágenes de prueba



(a) Imagen 1 con umbral 4 (b) Imagen 2 con umbral 4

(c) Imagen 3 con umbral 4 (d) Imagen 4 con umbral 4

Figura 4.19: Imágenes de prueba con umbral 4

eliminados es para las imágenes 2 y 4 que son visiblemente más heterogéneas. Parece lógico que en la 4.18c se eliminen siempre una

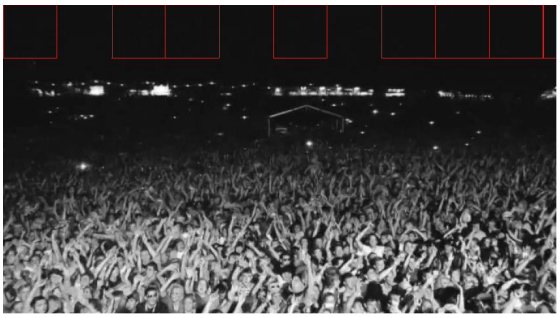
(a) Imagen 1 con umbral 2.5

(b) Imagen 2 con umbral 2.5

(c) Imagen 3 con umbral 2.5

(d) Imagen 4 con umbral 2.5

Figura 4.20: Imágenes de prueba con umbral 2.5



(a) Imagen 1 con umbral 1

(b) Imagen 2 con umbral 1

(c) Imagen 3 con umbral 1

(d) Imagen 4 con umbral 1

Figura 4.21: Imágenes de prueba con umbral 1

gran cantidad de píxeles, dado que gran parte de ella tiene un color negro y prácticamente en todos los umbrales se eliminan las mismas

regiones, con pequeñas variaciones.

A medida que decrece el umbral, con ello lo hace el porcentaje de píxeles eliminados. En 4.21 vemos que ni la imagen 4.21b ni la 4.21d se ven afectadas. Sin embargo, las restantes (4.21a y 4.21c) siguen presentando píxeles eliminados. En la primera el porcentaje decrece notablemente con respecto al que tiene en 4.19a, dejando atrás algunas regiones claramente homogéneas. El hecho de que la imagen 4.18c no presente apenas variación entre los tres umbrales es debido a que las zonas homogéneas que tiene, son de muy baja entropía dado a que son regiones totalmente negras, y prácticamente cualquier umbral las detectará como tal. El resto de la imagen es muy heterogénea, y por eso se necesita un umbral más alto para considerarla como zona de baja entropía.

Con este análisis, observamos que habrá un cierto grupo de imágenes como 4.18c que siempre podrán eliminar un alto número de vectores, independientemente del umbral, por lo que éste nunca será un porcentaje bajo, aunque sí lógico. En 4.19 observamos que en la figura 4.19a hay regiones eliminadas que no son tan homogéneas, como la zona de las luces superiores, aunque en 4.19d se elimina el cielo, que sí se puede tomar como tal. Podríamos considerar este umbral como el correcto, pero en el primer caso, se eliminaría más de un 36 % de los píxeles -que además sería inadecuado eliminar-, lo que implica más de un tercio de la imagen y ello quizá decrementaría la fiabilidad de nuestros descriptores. Además, como previamente se ha dicho en este apartado, es preferible eliminar en defecto que en exceso. Por esto y debido al porcentaje más bajo de píxeles eliminados en el caso de la figura 4.20, decidimos escoger como umbral final el valor 2.5.

5. Características de movimiento

Tras obtener el flujo óptico de todos los vídeos y filtrar las regiones de baja textura, eliminaremos un margen de 5 vectores correspondientes a los píxeles en los bordes, ya que si existe un movimiento, éstos píxeles pueden desaparecer de un frame a otro. Por lo tanto a la hora de buscar correspondencias en el siguiente fotograma, dado que no se encontrará en él, su estimación podría ser errónea. Posteriormente, de la información aportada por los valores de cada componente de los vectores se obtendrá información relacionada con el ángulo y módulo de éstos. Además, a raíz de los datos, se puede estimar también el movimiento de cámara presente (véase la figura 5.1).

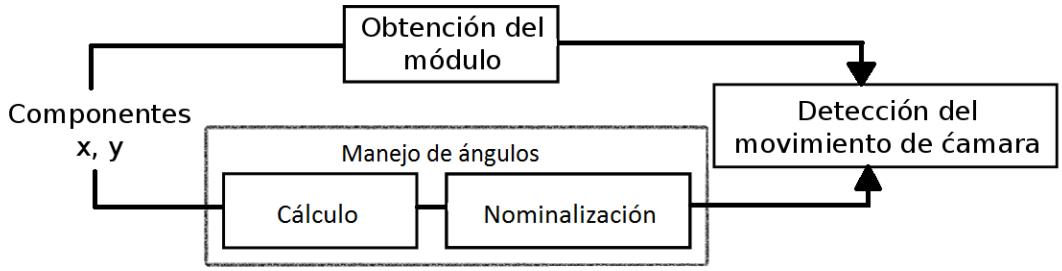


Figura 5.1: Proceso para obtener características de movimiento

5.1. Manejo de ángulos

Tras el procedimiento llevado a cabo en el apartado 4 y eliminar aquellas componentes de regiones homogéneas, tenemos los datos necesarios para calcular el ángulo del vector de movimiento correspondiente a cada píxel.

5.1.1. Puntos de referencia

Antes de determinar los ángulos, debemos establecer los puntos de referencia de los que partimos para ello. Siendo el centro el origen de coordenadas, se considera una x positiva cuando ésta está en la mitad derecha, y negativa si no lo está. Igualmente, una y será positiva en la mitad superior, y negativa en la inferior (ver figura 5.2).

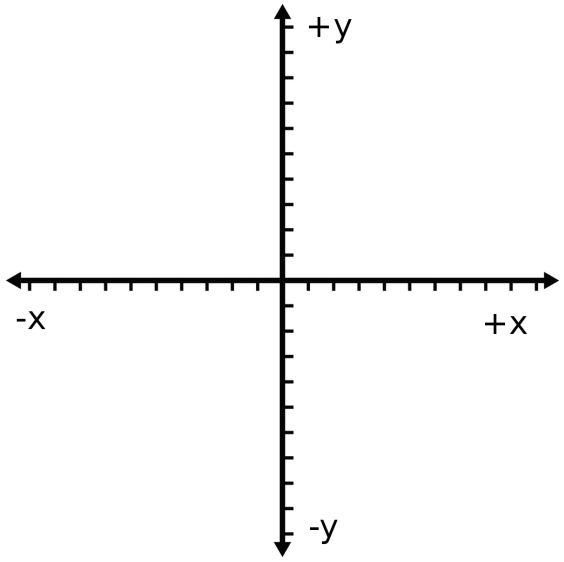
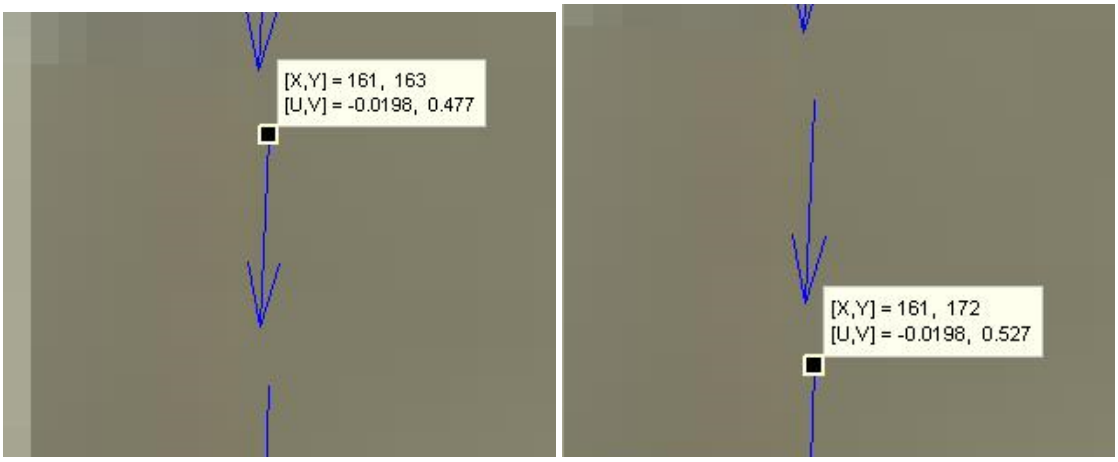


Figura 5.2: Eje de coordenadas y su signo

A la hora de interpretar las direcciones de los vectores del flujo óptico, se toma como punto de origen la esquina superior izquierda de la imagen, considerando como eje negativo de y todo aquello que quede por debajo. Sin embargo, tras calcular los ángulos y comprobar el resultado con el diagrama de flechas del flujo óptico, se descubre una incongruencia: la coordenada y tiene el signo erróneo y, sin embargo, está bien representada, como se puede ver en la figura 5.3a. En esta imagen X e Y representan las coordenadas en las que está dibujado el vector, U y V son los valores de las componentes x e y del valor del flujo óptico.

Como vemos, en este caso la componente y del vector es positiva, y según nuestro origen de coordenadas, debería ser negativa ya que apunta claramente hacia abajo. Esto dará problemas en un futuro, ya que los ángulos calculados se encontrarán siempre en el lado opuesto del eje x.



(a) Ejemplo del signo incorrecto de la componente y

(b) Ejemplo del valor creciente del eje de coordenadas

Figura 5.3: Demostración del valor incorrecto de y

Esta situación se da debido a que a la hora de dar valores a las coordenadas de la imagen para posteriormente dibujar los vectores, aunque se considera igualmente la esquina superior izquierda como origen, le hemos dado valores crecientes según el píxel se aleja hacia la parte inferior. Como vemos en la figura 5.3b, a pesar de que el punto marcado está por debajo del indicado en la figura 5.3a, el valor de la coordenada Y en la que se ha dibujado es mayor.

Ya que el flujo óptico en este punto ya está calculado, para solucionar este problema y que sea consistente a la hora de los cálculos con nuestro sistema de referencia elegido, simplemente se cambiará el signo de los valores de la componente y tras el cálculo del flujo óptico, con lo que se realizarán todas las operaciones posteriores.

5.1.2. Cálculo

Partiendo de las componentes x e y, como bien sabemos, se puede obtener la tangente del ángulo que formaría el vector:

$$\tan \phi = \frac{y}{x} \tag{5.1}$$

Y a su vez, partiendo de la tangente podemos obtener el propio ángulo en radianes:

$$\phi = \arctan(\tan\phi) = \arctan\left(\frac{y}{x}\right) \tag{5.2}$$

Sin embargo esto tiene un inconveniente: la arcotangente sólo dará ángulos de entre π y -π (ó $\frac{3\pi}{2}$), la mitad derecha de la circunferencia, independientemente de si el vector se encuentra ahí o no. Esto es debido a los signos de las componentes, cuya procedencia se desconoce en la tangente:

x	**y**	**tan**
+	+	+
+	-	-
-	+	-
-	-	+

Tabla 5.1: Combinación de signos en la tangente

Debido a lo que sucede en la tabla 5.1, cada valor de tangente tiene 2 ángulos posibles correspondientes con una diferencia de π radiantes entre ellos. Lo que realmente indica si es uno u otro, es la componente en x y por lo tanto esto se resuelve mediante la comprobación de su signo. En el caso de ser positivo, nuestro ángulo estará bien dado por la tangente ya que se encontrará en el lado derecho de la circunferencia. Por lo contrario, si la componente x es negativa, necesitaremos sumar o restar π radianes al ángulo obtenido, para así situar el correspondiente del lado izquierdo. Para tratar también exclusivamente con ángulos positivos, se comprueba si éstos son negativos y se les suma 2π como se refleja en la ecuación 5.3.

$$\phi = \begin{cases} \arctan(\frac{y}{x}) & \text{si } x > 0 \quad y < 0 \\ \arctan(\frac{y}{x}) + \pi & \text{si } x < 0 \\ \arctan(\frac{y}{x}) + 2\pi & \text{si } x > 0 \quad y < 0 \end{cases} \tag{5.3}$$

5.1.3. Nominalización

Dar como dato del vector su ángulo, ya sea en radianes o grados, puede ser complicado de interpretar debido a su caracter circular y su rango de valores positivos y negativos de 0 a ∞. Es por esto que se decide agrupar los ángulos en ocho puntos cardinales. Para ello, se divide la circunferencia en 8 partes, obteniendo así amplitudes de

$\frac{\pi}{4}$ y situamos las divisiones tal y como se muestran en las figuras la imagen 5.4 y en la tabla 5.2.

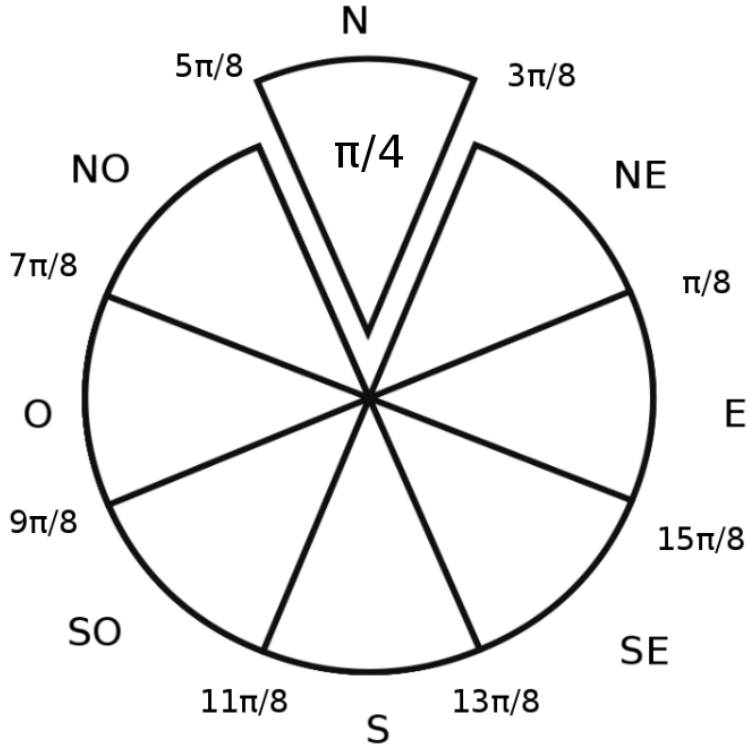


Figura 5.4: Representación de la asignación de ángulos

Punto cardinal	Ángulos
E	$\frac{15\pi}{8}$ - $\frac{\pi}{8}$
NE	$\frac{\pi}{8}$ - $\frac{3\pi}{8}$
N	$\frac{3\pi}{8}$ - $\frac{5\pi}{8}$
NO	$\frac{5\pi}{8}$ - $\frac{7\pi}{8}$
O	$\frac{7\pi}{8}$ - $\frac{9\pi}{8}$
SO	$\frac{9\pi}{8}$ - $\frac{11\pi}{8}$
S	$\frac{11\pi}{8}$ - $\frac{13\pi}{8}$
SE	$\frac{13\pi}{8}$ - $\frac{15\pi}{8}$

Tabla 5.2: Asignación de ángulos a los cardinales

5.2. Obtención del módulo

Una vez calculado el ángulo que forman las componentes de cada vector de movimiento de los píxeles, es interesante calcular su módu-

lo mediante la ya conocida fórmula 5.4, siendo x e y los valores de cada componente de un píxel en concreto.

$$|v| = \sqrt{x^2 + y^2} \tag{5.4}$$

Este valor nos indicará si el movimiento es más o menos brusco, dado que cuanto más grande sea el valor del módulo de un píxel, implicará que éste se ha movido más distancia y, por lo tanto, su movimiento es más rápido que el de otro píxel con menos módulo.

5.3. Detección de movimiento de cámara

Existe en la literatura trabajos que establecen una relación entre la dirección del frame y el movimiento de cámara canónico más presente en un plano o un vídeo. A esto se le llama *camera motion estimation* o estimación del movimiento de cámara. En una primera aproximación, nos ayudaremos de la estimación de movimiento de cámara con técnicas de parametrización del movimiento.

Mediante la extracción del flujo óptico, se pueden obtener parámetros que definan el movimiento a partir de las componentes x e y de los vectores. El objetivo es que, a partir de estos parámetros y las coordenadas de los píxeles, podamos obtener una estimación del movimiento.

Consideramos una matriz U de tamaño $Nx3$ siendo N el número de píxeles del fotograma y cuyas dos primeras columnas corresponden a las coordenadas x e y del vector del flujo óptico en ese píxel, tal y como se expresa a continuación:

$$U = \begin{pmatrix} x(1) & y(1) & 1 \\ x(2) & y(2) & 1 \\ \vdots & \vdots & \vdots \\ x(N) & y(N) & 1 \end{pmatrix}$$

Como se puede ver en la ecuación 5.5, U debe ser igual al producto de la matriz de coordenadas de los píxeles (normalizadas entre -1 y 1 situando el origen en el centro de la imagen) a la que llamaremos A, con la matriz de parámetros P que estamos buscando.

$$U = AP \tag{5.5}$$

Dado que la matriz U ya la tenemos porque hemos calculado previamente las componentes de los vectores de flujo óptico, y también conocemos la matriz de las coordenadas de los píxeles, A, podemos simplemente despejar y obtenemos la fórmula 5.6:

$$P = A^{-1}U \tag{5.6}$$

Tenemos así una matríz P de tamaño 3x3, que representan los 9 parámetros del movimiento, siendo el último siempre un 1.

$$P = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix}$$

Habiendo conseguido ya la matriz que se buscaba, deberíamos poder proceder a llevar a cabo alguno de los métodos ya existentes en trabajos como [14] y [15]. Sin embargo, la parametrización llevada a cabo en [14] es diferente, por lo cual estos métodos no son aplicables en nuestro caso. Además, en [15] se utiliza una base de datos previamente etiquetada con el fin de determinar el umbral para el que se deba considerar la presencia de un movimiento de cámara concreto, y nosotros carecemos de una base de datos con este tipo de etiquetado.

Aunque hemos comprobado que no podremos llevar a cabo la tarea mediante estos métodos, dado que ya se ha obtenido la matriz P, se reutiliza para sacar de nuevo la matriz U de la que se hablaba previamente. Con ello obtendremos una aproximación del flujo óptico, como se ve en la imagen 5.5.

Aunque esta representación no indica correctamente el flujo óptico, sí podría ser un indicativo del movimiento general de los píxeles en la imagen y, si así fuera, la simple determinación del ángulo de éstos nos ayudaría a detectar el movimiento de cámara aplicado. Sin embargo, tras la representación gráfica de los vectores con este método en algunos vídeos, se comprueba que la relación que existe entre la dirección de los vectores y el movimiento de cámara no se ajusta a los objetivos buscados, por lo que se descarta este método.

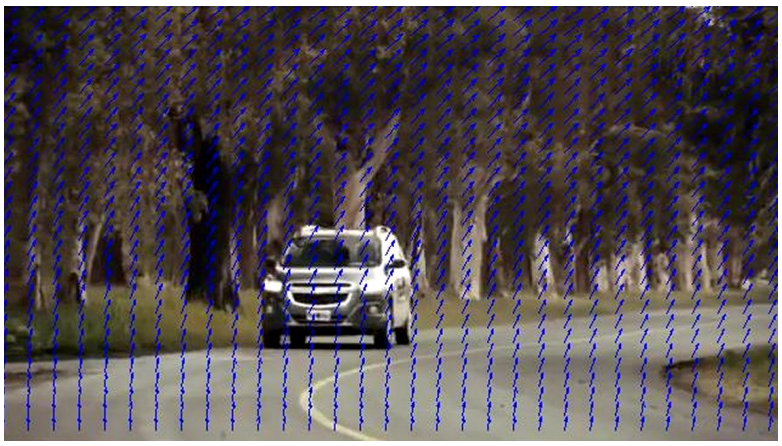


Figura 5.5: Flujo óptico estimado con el uso de la matriz P

Debido a que las soluciones mediante parámetros de movimiento no son aplicables en este caso, se decide recurrir a un análisis de las direcciones de los píxeles en los márgenes de la imagen. Esto es debido a que el centro de la imagen existe más probabilidad de que se encuadre un sujeto y, por lo tanto, de que haya movimiento independiente de la cámara.

Existen varios tipos de movimiento de cámara como se puede ver en [20]. Sin embargo, sólo se detectarán los más comunes:

- **Pan:** es aquél que implica un movimiento de la cámara en horizontal, ya sea hacia izquierda o derecha. Si éste movimiento fuese hacia la derecha, los píxeles de los márgenes se moverán a hacia la izquierda, y viceversa.

- **Tilt:** es equivalente al pan pero en el eje Y, implica movimientos verticales de cámara. De esta manera, cuando hay un tilt hacia arriba de la cámara, los píxeles se moverán hacia abajo y viceversa.

- **Zoom:** corresponde a un acercamiento o alejamiento de la imagen en el eje Z. En el primer caso, los píxeles de los bordes se moveran hacia el exterior y en el segundo hacia el interior.

- **Cámara fija:** en este caso existen dos variantes, ya que la cámara puede estar fija en un trípode o puede ser cámara al hombro, lo cual implicará cierto movimiento. En este caso, el módulo del movimiento de los píxeles será nulo o muy pequeño.

Sabiendo esto, se seguirá el proceso representado en el diagrama de flujo de la figura 5.7 para determinar el movimiento de cámara, que se estima a partir de los valores cardinales y modulares de los vectores de todos los píxeles de cada frame. Antes de comenzar, se escogerá un 10 % en cada uno de los lados de la imagen, con respecto a la altura y anchura. Esto es, si la imagen tiene una dimensión de 1280x720 píxeles -tras la eliminación de la estimación de los bordes-, los márgenes horizontales tendrán un tamaño de 128 píxeles cada uno, y los verticales de 72 (ver 5.6).

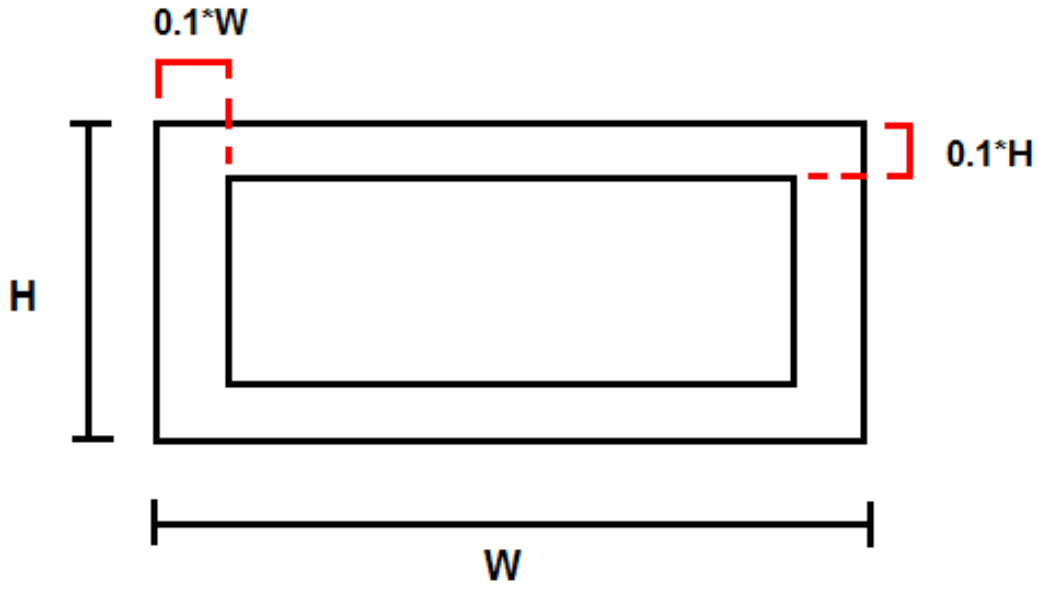


Figura 5.6: Asignación del tamaño de los márgenes

En los siguientes apartados se explicarán las funciones de cada uno de los parámetros que sirven de ayuda en la tarea de detección de movimiento de cámara, así como el proceso para la determinación de sus umbrales de decisión. Los procesos de fijación de los umbrales será manual, debido a que carecemos de una base de datos etiquetada previamente con los movimientos de cámara, por lo que no podemos llevar a cabo un proceso de aprendizaje máquina.

5.3.1. Detección de cámara fija

Para detectar una situación de cámara fija utilizaremos la información aportada por el cálculo de los módulos de los vectores. El objetivo aquí es determinar una cifra relacionada con el módulo que

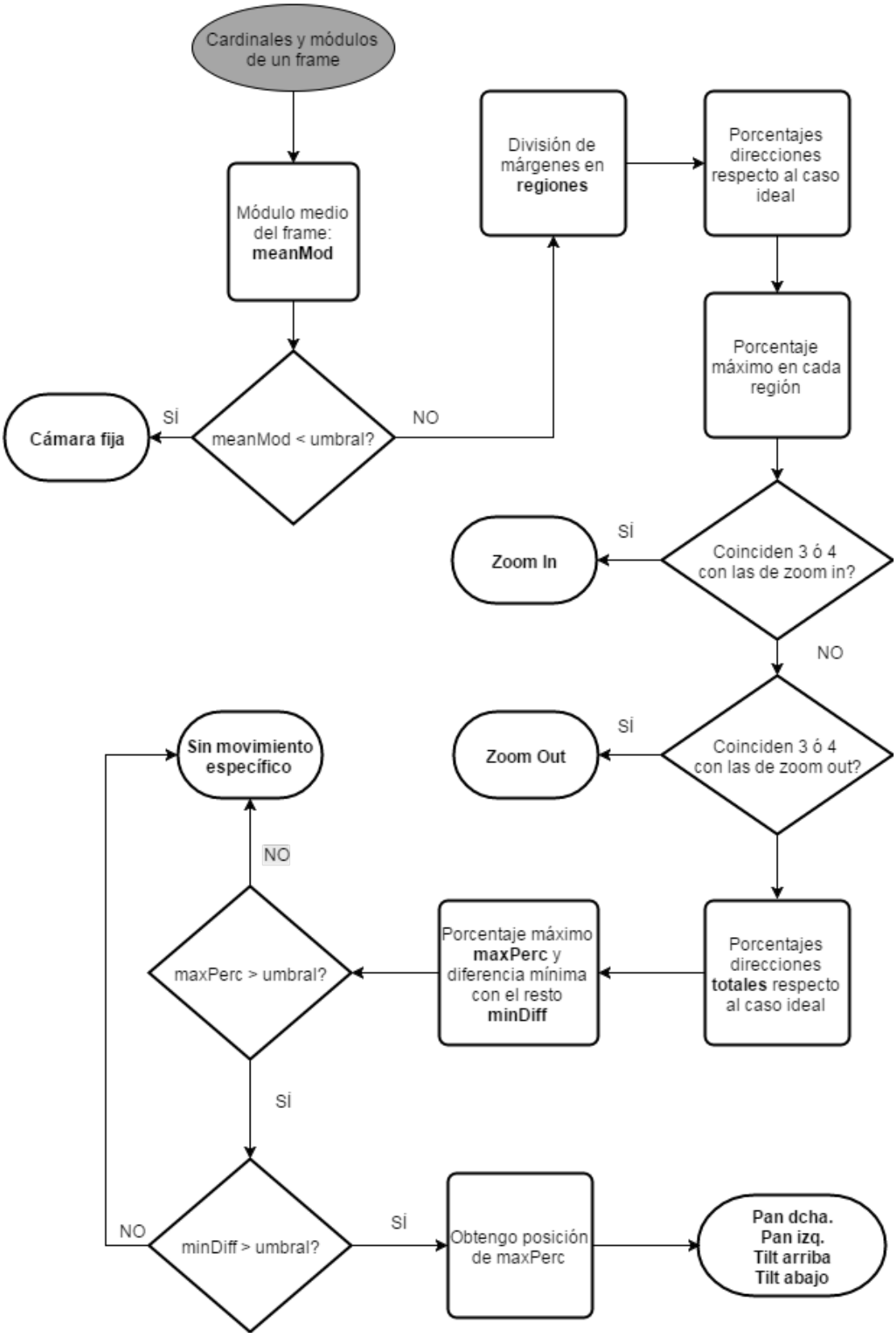


Figura 5.7: Diagrama de flujo sobre la detección de movimiento de cámara

represente a todo el frame, así como un umbral para éste valor que nos ayude a decidir sobre si el frame está hecho con una cámara fija, como en el ejemplo de la imagen 5.8.

Figura 5.8: Flujo óptico en una toma de cámara fija

Aquí se puede observar que en los bordes los vectores son nulos o muy pequeños, por lo que calculamos el módulo de las componentes x e y de los vectores y los sumamos, para posteriormente hacer su media dividiéndolo por el número total de píxeles que se han tenido en cuenta. Así obtenemos el parámetro *meanMod* que en este caso vale 0.0857. Tras comprobar más frames fijos en el caso de este vídeo en particular, se descubre que los valores no superan 0.12. Para comprobar si este valor es útil en otros casos, se escogen frames de otros vídeos que también son fijos, pero con vectores menos claros, como el mostrado en la imagen 5.9.

En esta imagen existen vectores con gran módulo debido a que parte del movimiento está situado en los bordes, pero sin embargo hay regiones que no están en movimiento y que pertenecen al fondo de la imagen, por lo que demuestra que la cámara está fija. El umbral que busquemos debe englobar también este tipo de casos en los que encontramos movimiento en la zona de márgenes, pero cuya cámara sea fija aun así. En este caso, el valor de la media de los módulos es de 0.4441, por lo que fijamos un umbral de similar valor como es el de 0.45. Sabemos que este umbral englobará también imágenes como la mostrada en 5.8, pero necesitamos comprobar que también excluye a aquellas que realmente sí tienen movimiento de cámara (figura 5.10a)

Figura 5.9: Toma de cámara fija con vectores poco claros

o cuya cámara al hombro es lo suficientemente inestable como para no considerarlo cámara fija (figura 5.10b).

(a) Cámara no fija (b) Cámara al hombro

Figura 5.10: Flujo óptico en cámaras sin trípode fijo

En la primera figura obtenemos un valor de 0.661, por lo cual será considerada como cámara no fija, lo cual es correcto. En la segunda su valor es de 0.6014, por lo cual tampoco será considerada como tal. Aunque en este caso pueda parecer incoherente, en la visualización del vídeo se observa que esta toma de cámara al hombro es muy inestable, por lo que es lógico no considerarla fija. Es por esto que la detección de cámara fija va a depender además, de si lo es debido a que se ha utilizado un trípode fijo, o si es cámara al hombro. Esto último implicará que, a pesar de que se puede considerar teóricamente que estamos ante una toma fija, el plano tendrá cierto movimiento que se verá reflejado en el módulo de los vectores de flujo óptico. Dependiendo de lo suave o brusco que sea el movimiento producido por la cámara al hombro, detectaremos este tipo

como una cámara fija o no. De igual manera esto sucederá también en el caso de los zooms muy sutiles, pero dado que existen pocos en la base de datos, no será algo determinante en la obtención de resultados.

5.3.2. Asignación de pesos a las direcciones

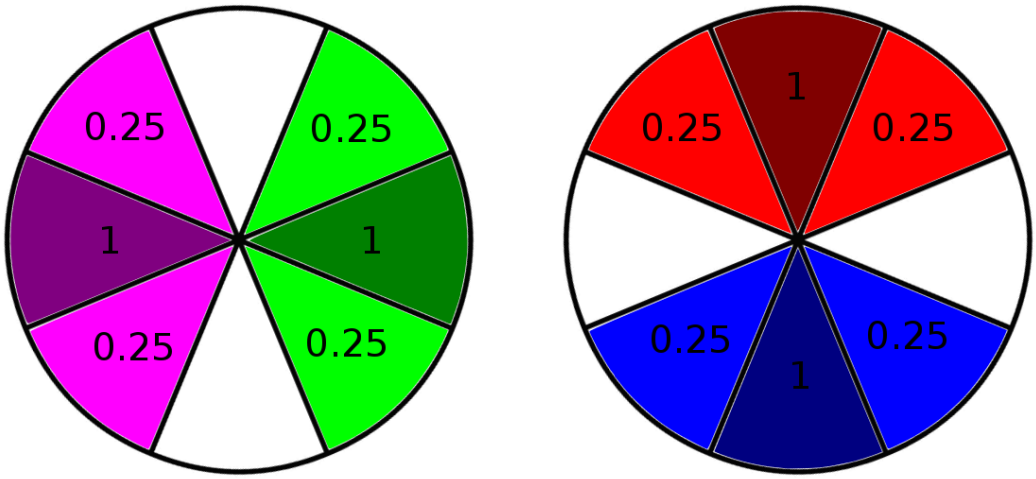
En este punto del trabajo, ya conocemos los valores de dirección cardinal de los vectores de movimiento en los bordes de la imagen, de los cuales podremos extraer el movimiento de cámara del frame. Además, tras escoger el valor del umbral comentado en el apartado 5.3.1, habremos descartado aquellos fotogramas con cámara fija y sólo trabajaremos con el resto.

Mediante el visionado de los vídeos con la representación de los vectores se puede ver que los movimientos de cámara más claros tienen una mayoría de ángulos apuntando en un único punto cardinal. Sin embargo, debido a ciertas variaciones o a que el movimiento no es tan "puro", estos vectores pueden variar ligeramente y pertenecer a un cardinal contiguo, por lo que no son una prueba clara de que la imagen se mueva en la dirección concreta que buscamos, pero siguen siendo datos relevantes debido a su proximidad a la dirección buscada. Por ejemplo, si tuviésemos un tilt hacia abajo pero no de manera totalmente vertical, sino con cierta inclinación, los vectores apuntarían hacia arriba pero inclinados hacia la dirección totalmente opuesta del ángulo de movimiento. Esto no quiere decir por ello que no sea un tilt -lo es, aunque no en el sentido estricto, pero aún así nos interesa detectarlo-, por lo que se han de tener en cuenta también ciertos rangos del ángulo en cada caso.

A la hora de asignar pesos lo que se pretende es dar más valor a aquellas direcciones que tendrían los movimientos "puros" (N, S, E, O), para que de esta manera, si ante un movimiento puro existen otros vectores en direcciones "diagonales" (NE, NO, SE, SO) éstos no lleguen a ser más determinantes a la hora de estimar el movimiento, pero sí que faciliten la labor de decisión.

En la figura 5.11, que tiene correspondencia con los cardinales asignados en la figura 5.4, se puede ver la asignación de pesos a cada dirección cardinal. En 5.11a los vectores situados en la región

fucsia serían los correspondientes a un pan a la derecha (los píxeles se mueven hacia la izquierda) y por el contrario, los situados en la región verde, corresponderían con un pan hacia la izquierda (los píxeles se mueven hacia la derecha). Asimismo, en la figura 5.11b se representan en color azul las regiones de tilt hacia arriba (píxeles hacia abajo) y en rojo los correspondientes a un tilt hacia abajo (píxeles hacia arriba).



(a) Asignación según pan (b) Asignación según tilt

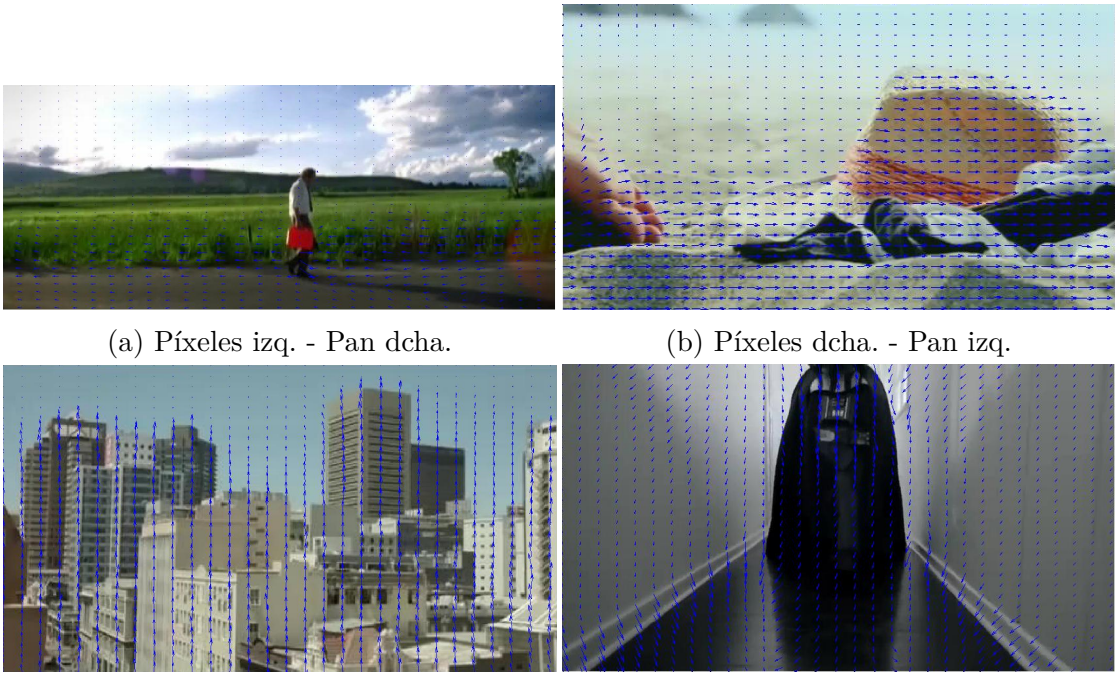
Figura 5.11: Asignación de pesos a los cardinales

Tras la aplicación, multiplicamos cada número de píxeles en cada dirección por su valor correspondiente en cada uno de los 4 casos. Este valor dividido entre el número de píxeles y multiplicado por 100, nos dará un porcentaje de similitud con respecto al caso ideal de cada movimiento. Esto formará un vector de 4 posiciones que contienen el valor correspondiente al pan izquierdo, pan derecho, tilt hacia arriba y tilt hacia abajo, por ese orden.

Eficacia en movimientos comunes

El objetivo es determinar la eficiencia de este valor a la hora de indicar uno u otro movimiento. Para esto, se escogen 4 imágenes diferentes (ver imágenes en 5.12), cada una con un tipo de movimiento de pan o tilt. Tras aplicar pesos a las direcciones cardinales de sus

márgenes, obtenemos la tabla 5.3. En ella se representan los porcentajes obtenidos respecto a los casos ideales de cada movimiento N-S-E-O. Estos valores indican si los píxeles se mueven en mayor o menor medida a un lado u otro.



(a) Píxeles izq. - Pan dcha.

(b) Píxeles dcha. - Pan izq.

(c) Píxeles arriba - Tilt abajo

(d) Píxeles abajo - Tilt arriba

Figura 5.12: Imágenes con pan y tilt

	% derecha	% izquierda	% arriba	% abajo
Pan dcha.	2.6427	22.4652	6.8322	0.2723
Pan izq.	31.0743	0	0.6594	2.3527
Tilt abajo	0.6446	0.4537	33.914	0
Tilt arriba	0.6678	5.1335	0	18.7654

Tabla 5.3: Porcentajes para cada imagen en 5.12

En la tabla 5.3 se puede comprobar que los resultados tienen sentido y que sería posible una futura detección en base a esos valores. En la imagen 5.12a obtenemos un mayor número de píxeles hacia la izquierda, mientras que en comparación el resto son muy pequeños. Más claro está en 5.12b y 5.12c en los que se llegan a obtener más de un 30 % de movimiento en la direcciones predominantes de los píxeles, y un 0 % en aquellas opuestas. A su vez, 5.12d obtiene un porcentaje alto de píxeles hacia arriba, pero también nulo en la dirección contraria. Esto implica que, efectivamente, si los movimientos

de pan y tilt son muy predominantes, obtendremos un gran número de movimiento de píxeles en dirección contraria al movimiento de la cámara, por lo cual será fácil diferenciarlos.

Este mecanismo de asignación de pesos se utilizará, por tanto, para la detección de pan y tilt como para la detección del zoom, como veremos próximamente en los apartados 5.3.3 y 5.3.4.

5.3.3. Detección de zoom

En el momento de detectar el zoom de un frame, necesitamos diferenciar las regiones de los márgenes en los que se encuentran los diferentes ángulos. Según las direcciones de estas zonas, consideraremos zoom in, zoom out o ninguno (ver imagen 5.13 y tabla 5.4).

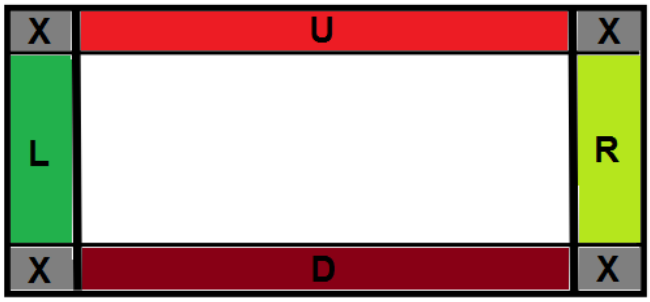


Figura 5.13: Diferentes regiones del margen

	U	D	L	R
Zoom In	Arriba	Abajo	Izquierda	Derecha
Zoom Out	Abajo	Arriba	Derecha	Izquierda

Tabla 5.4: Relación de zoom con los ángulos de las regiones

Según la imagen 5.13, no se han tenido en cuenta aquellas regiones de solape entre dos diferentes (las denominadas como X), ya que aun dividiendo estas regiones en diagonal para asignarlas a una u otra zona, éstas aportarían pocos píxeles. Además, si estuviésemos en el caso de un zoom, estas zonas serían poco significativas a la hora de determinar el ángulo predominante, ya que serán vectores diagonales y aportarían un peso muy bajo.

En este caso seguiremos el procedimiento explicado en 5.3.2 para obtener el vector de cuatro posiciones en cada una de las cuatro regiones diferentes. Posteriormente, obtendremos el mayor porcentaje en los cuatro casos, teniendo así cuatro valores que nos indicarán a qué dirección corresponde el movimiento de cada zona. De esta manera, y de forma equivalente a la tabla 5.4, para declarar cada tipo de zoom, las direcciones mayoritarias de cada región deben ser las de la tabla 5.5.

	U	D	L	R
Zoom In	N	S	O	E
Zoom Out	S	N	E	O

Tabla 5.5: Direcciones con porcentaje máximo de cada región

Ya que el cumplimiento de estas 4 condiciones a la vez es complicado incluso cuando hay un zoom debido a fallos de estimación u objetos situados en el margen, se establece que no sea necesario cumplirse todas ellas para considerar que estamos ante un zoom. El hecho de que se cumplan 3 de ellas de manera simultánea es ya lo bastante exclusivo como para determinar este movimiento. Si estas condiciones no se cumplen, pasamos a detectar si nos encontramos ante un pan o un tilt.

5.3.4. Detección de pan y tilt

Tras descartar las situaciones de cámara fija o zoom, debemos comprobar si nos encontramos ante un pan o tilt. Para esto, a diferencia del zoom, no necesitamos hacer diferencia entre las regiones del margen, ya que todos sus píxeles necesitarán ir en una dirección u otra.

De nuevo aplicamos el método descrito en el apartado 5.3.2 a lo largo de todo el margen para obtener el vector de cuatro posiciones, y detectamos su máximo, que nos dará el valor *maxPerc*. En este caso, a diferencia también del zoom, necesitamos observar este valor y compararlo con el resto de los obtenidos, porque pueden darse dos situaciones:

- Que el valor máximo sea de por sí muy pequeño, lo que demostrará que el patrón de los bordes no corresponde con los que se

han contemplado, por lo que no hay un movimiento muy claro.

- Que el valor máximo sea muy similar a, al menos, uno de los demás valores obtenidos. Aquí se puede ver que realmente ninguno de ellos predomina suficientemente. En este caso podría ser que tuviésemos una combinación de movimientos, lo cual no nos interesaría deducir.

Si se da una de estas situaciones, o ambas, determinaremos que el movimiento de cámara no será específico. De nuevo nos encontramos con una situación que necesita el establecimiento de algún umbral para la decisión. En el primer caso determinaremos un valor mínimo para el *maxPerc*, llamado *minValue*. En el segundo caso, calcularemos la diferencia mínima que hay entre el máximo valor de ese vector de cuatro posiciones y el resto de porcentajes, eligiendo para ella umbral. De esta manera nos aseguraremos que el siguiente valor con respecto a nuestro *maxPerc* sea lo suficientemente pequeño con respecto a este como para considerar un movimiento predominante. El proceso llevado a cabo para la selección de estos valores se explicará más a fondo en el apartado 5.4.2.

5.4. Funcionamiento a nivel de plano

Una vez detectado el movimiento de cámara en cada frame, se puede proceder a obtener el movimiento de cada plano ya que es más perceptible al ojo humano que el de un único fotograma. Por ello primero se procede a detectar cuándo hay un cambio de plano en los vídeos y posteriormente se comprueba el funcionamiento del trabajo previo.

5.4.1. Detección de cambios de plano

Para esto, utilizaremos como guía el trabajo ya realizado en [7], ya que en este artículo se ha utilizado la misma base de datos y ya se determinó el umbral necesario para una óptima detección de planos.

En [7] se basan en la diferencia que existe entre el último fotograma de un plano y el primero del siguiente. Primero se obtiene la

Sum of Absolute Differences (SAD) para cada par de frames consecutivos, esto es, la diferencia de los valores de la intensidad de los píxeles acumulada a lo largo de la imagen, en valor absoluto y dividida entre el número de píxeles (ver ecuación 5.7).

$$D(n) = \frac{1}{H \times W} \sum_{x=1}^{W} \sum_{y=1}^{H} |I_n(x,y) - I_{n-1}(x,y)| \tag{5.7}$$

Posteriormente se calculan la primera y segunda derivada para conocer la variación de la *SAD* y poder detectar sus máximos. Considerando $D'(n) = D(n) - D(n-1)$, la segunda derivada se corresponderá con la ecuación 5.8. Es aquí donde el umbral escogido para determinar si ha habido o no un cambio de plano es que el valor de M sea superior a 0.18.

$$M(n) = -D''(n+1) = -(D'(n+1) - D'(n)) \tag{5.8}$$

Una vez obtenidos los tipos de movimiento en cada frame y los planos del vídeo, necesitamos establecer algún valor que nos haga determinar cuál es el movimiento predominante en ese plano. Por ello, en este caso lo que hacemos es calcular el tipo de movimiento más frecuente en los frames, es decir, la moda. Posteriormente, para asegurar que el plano no es una mezcla de diferentes movimientos, calculamos el porcentaje de frames que tienen el movimiento moda, *percMode*. Si éste es superior a un umbral, consideraremos que ese es el movimiento predominante del plano. De lo contrario, le asignaremos la categoría de movimiento no específico.

5.4.2. Selección de umbrales

Una vez explicados los apartados previos obtenemos varios umbrales a determinar para aumentar la eficacia de la detección de movimiento:

- minDiff: la diferencia mínima que ha de haber entre el valor máximo del vector y el resto de los cuatro valores obtenidos tras la asignación de pesos (ver apartado 5.3.2).
- minValue: una vez obtenido el máximo valor del vector descrito en el apartado 5.3.2, debe cumplir ser mayor que cierto número.

- percMode: tras calcular el movimiento de cada frame, se obtiene el aquél más presente a lo largo de los fotogramas del plano y se establece qué porcentaje de frames lo tienen. Este umbral determinará el porcentaje necesario para considerar si el movimiento moda de los frames será también el predominante del plano o no.

Dado que la base de datos no está etiquetada respecto a movimientos de plano ni de frame, el análisis para determinar estos valores es de nuevo visual, realizando comparativas entre lo que percibe la lautora y lo que se obtiene del método utilizado.

minDiff	3
minValue	7
percMode	45

Tabla 5.6: Parámetros en detección de movimiento de cámara

Por motivos de espacio, no se pondrán aquí los resultados de cada una de las variaciones de parámetros que se han comprobado. En ese caso, sí se proveerán algunos resultados obtenidos mediante la fijación de parámetros a los valores indicados en la tabla 5.6, los cuales se utilizarán de ahora en adelante.

Para realizar las pruebas, previamente se han etiquetado uno a uno los movimientos que parecen predominantes en cada plano de 7 vídeos con características muy diferentes, buscando la presencia de cada uno de los movimientos a detectar, así como los casos ambiguos como pueden ser una cámara al hombro brusca o la combinación de varios movimientos. Cabe aclarar que aunque esta asignación es manual, es complicado detectar de manera objetiva qué movimiento está más presente en una combinación de varios movimientos de cámara ya que es imposible de manera visual cuantificar cuántos píxeles se mueven hacia un lado u otro. Sin embargo, esto sí se puede obtener tras el análisis del vídeo, ya que nos dará una medición sobre cuál de ellos está en más cantidad. Por eso, en los casos más ambiguos se considerará visualmente como movimiento no específico, aunque quizá automáticamente se obtenga un movimiento predominante. La terminología usada para cada tipo de movimiento se corresponde con aquella indica en la tabla 5.7.

0	No específico
1	Pan izquierda
2	Pan derecha
3	Tilt abajo
4	Tilt arriba
5	Cámara fija
6	Zoom in
7	Zoom out

Tabla 5.7: Correspondencia nominal de movimiento

En la tabla 5.8 se tiene un vídeo observado con un movimiento generalmente fijo y sin zoom. En algunos casos de cámara fija encontramos un fallo de detección, posiblemente debido a que la cámara es llevada al hombro y hay algún tipo de movimiento lo suficientemente grande como para no ser considerado fijo. También podemos observar que el plano 16 considerado visualmente con movimiento no específico, se detecta como un movimiento de pan hacia la izquierda. Esto podría deberse a que a lo mejor el plano tiene una combinación de diferentes movimientos, siendo éste uno de ellos.

En el vídeo utilizado para realizar la tabla 5.9 encontramos varios planos etiquetados como movimiento no específico, dado que en algunos casos hay combinación de movimientos y en otros directamente, es difícil de saber qué movimiento de los estipulados es. Debido a esto, únicamente presenta un acierto del 58 %, dado que aunque visualmente es difícil establecer cuál de los movimientos influye más, computacionalmente sí es posible establecerlo debido a los pesos dados a los cardinales (ver 5.3.2). Así, los fallos a la hora de la detección de un zoom, como se ha referido previamente, podrían ser provocados por la sutileza de éste, lo que lleva a que sea detectado como fijo. Sin embargo, dada la dificultad de la detección de estos y su poca presencia en la base de datos respecto al movimento fijo, no influirá en gran medida.

La tabla 5.10 detecta bien todos los planos, debido a que tienen movimientos muy claros. Sin embargo, en el plano número 4 obtenemos un fallo. Esto es debido a que se considera de nuevo que el plano no tiene movimiento específico, ya que éste específicamente tiene un zoom in con tilt hacia arriba, lo que provoca que si no se detecta ningún movimiento específico, se detecte el más predominante, en

este caso según el cálculo, será el tilt.

Tras comparar el funcionamiento de este método de detección de movimiento de cámara con respecto a los observados, se puede ver que no es una tarea trivial. La clasificación manual utilizada a nivel de plano tiene un cierto grado de subjetividad, ya que en los casos de movimiento combinado los planos se podrían etiquetar de un modo u otro dependiendo de la persona que lo visualiza. Sin embargo, dado que estamos detectando el movimiento predominante mediante el análisis de los vectores de flujo óptico, el movimiento etiquetado puede ser uno que visualmente no se perciba en tanta medida. Un ejemplo de esto podría ser un plano que combine un pan a la derecha con un tilt hacia arriba, ya que habrá usuarios para los que destaque más el tilt que el pan y viceversa, aunque a la hora de detectar el movimiento se tendrán en cuenta factores más objetivos y se detectará aquél cuyas direcciones aporten más peso, como se explicó en el apartado 5.3.2, o en el caso de ser direcciones con similar grado de presencia, no se detectará ningún movimiento.

Núm. Plano	Mov. observado	Mov. obtenido	Acierto/Fallo
1	1	1	A
2	5	5	A
3	5	0	F
4	1	1	A
5	5	5	A
6	5	5	A
7	5	5	A
8	5	5	A
9	5	5	A
10	1	0	F
11	1	3	F
12	1	1	A
13	0	0	A
14	5	5	A
15	0	0	A
16	0	1	F
17	0	0	A
18	5	5	A
19	5	5	A
20	5	5	A
21	5	5	A
22	5	3	F
23	0	0	A
Acierto			82.6 %

Tabla 5.8: Detección de movimientos de cámara por plano en el vídeo "$0hdZoUqLV\_Q$".

Núm. plano	Mov. observado	Mov. detectado	Acierto/Fallo
1	6	5	F
2	4	4	A
3	0	0	A
4	2	2	A
5	4	4	A
6	1	1	A
7	0	4	F
8	0	1	F
9	7	7	A
10	0	1	F
11	0	1	F
12	1	1	A
13	5	5	A
14	0	4	F
15	7	5	F
16	7	7	A
17	5	5	A
% Aciertos			58 %

Tabla 5.9: Detección de movimientos de cámara por plano en el vídeo "$LSL147vLc8S$".

Núm. plano	Mov. observado	Mov. detectado	Acierto/Fallo
1	2	2	A
2	1	1	A
3	2	2	A
4	0	4	F
5	1	1	A
6	5	5	A
7	0	0	A
8	5	5	A
9	1	1	A
10	1	1	A
11	0	0	A
12	5	5	A
% Acierto			91 %

Tabla 5.10: Detección de movimientos de cámara por plano en el vídeo "$fhTW4oz - g7A$".

6. Extracción de descriptores

Tras llevar a cabo todos los pasos explicados en el capítulo 5, se obtienen una serie de medidas básicas en bruto de los vídeos, las cuales se pueden traducir en descriptores con significado de relevancia para la evaluación de la estética.

6.1. Descriptores estadísticos

A la hora de obtener descriptores a nivel de vídeo, parece lógico que un primer acercamiento sea el cálculo de medidas estadísticas. Esto se hára partiendo de los datos obtenidos a nivel de frame para posteriormente operar con ellos y obtener descriptores respresentativos de todo el vídeo.

6.1.1. Estadística circular

Debido a la utilización de valores circulares, el cálculo de la media y la desviación estándar no serán iguales que en el caso de los valores lineales. Un ejemplo de esto es que, si tenemos un ángulo de 30° con otro de -30° (ó 330°), la media nos dará un ángulo de 180°, el cual apunta en dirección completamente opuesta a la de los ángulos que tenemos, por lo que no es correcto. Esto se debe a que el cálculo de la media depende del origen de coordenadas y de la dirección de movimiento y dado que la desviación estándar depende de la media, ésta también tendrá el mismo problema.

Como solución, existe la *estadística circular*, que proporciona métodos como los que se explican en [10] para calcular la media y desviación típica de los ángulos. Este método consiste en transformar cada ángulo ϕ en un número complejo (ver fórmula 6.1) y posteriormente proceder al cálculo de la media aritmética como has-

ta ahora hacíamos: sumando todos los vectores y dividiendo por el número de ellos (ver fórmula 6.2):

$$z = \cos(\phi) + i\sin(\phi) \tag{6.1}$$

$$\bar{\rho} = \frac{1}{N}\sum_{i=1}^{N} z = \frac{1}{N}\left(\sum_{i=1}^{N}\cos(\phi) + i\sum_{i=1}^{N}\sin(\phi)\right) \tag{6.2}$$

Como $\bar{\rho}$ es un número complejo, podemos obtener el ángulo que éste forma de la misma manera que indicamos en el apartado 5.1, obteniendo así el ángulo medio (ver fórmula 6.3).

$$\bar{\phi} = \arg(\bar{\rho}) \tag{6.3}$$

Se sabe además que $\bar{\rho}$ se puede expresar mediante una exponencial compleja de ángulo $\bar{\phi}$ y módulo $\bar{R} = \sqrt{\left(\frac{1}{N}c\right)^2 + \left(\frac{1}{N}s\right)^2}$, siendo c el sumatorio de los cosenos de, s el de los senos de cada ángulo y N el número de ángulos. Con la ayuda de $\bar{R}$ se podrá determinar la desviación estándar, pues se cumple la ecuación 6.4:

$$\bar{S} = \sqrt{-2\ln\bar{R}} \tag{6.4}$$

Una vez conocemos cómo calcular la media y desviación estándar de un conjunto de ángulos, podemos proceder a la extracción de los descriptores que deseamos, que serán a nivel de vídeo. Aunque podría parecer lógico obtener primero la media y desviación típica de cada uno de los fotogramas, no nos será útil a la hora de obtener valores a nivel de vídeo. Esto se debe a que la media de los ángulos de todo el vídeo no se corresponde con la media de las obtenidas a nivel de frame, al igual que pasa con la desviación típica. Por lo tanto, a nivel de fotograma la única medida estadística que nos será de ayuda será la moda, ya que sí podremos obtener un valor con sentido cuando conozcamos el cardinal más presente de entre todos los que aparecen en cada fotograma.

6.1.2. Cálculo secuencial

Para poder calcular de manera fiable descriptores estadísticos a nivel de vídeo, se deben agrupar los valores de todos los píxeles de

cada frame en una sola matriz, y operar con ello. Si tenemos en consideración el tamaño de la matriz de ángulos de cada frame, que pueden llegar a ser dimensiones superiores a 1280x720, multiplicado esto a su vez por su número de fotogramas, que llegan a ser más de 1500, obtendremos matrices con varios cientos de millones de píxeles en el peor de los casos. Dado que en la realización de todos estos cálculos se utiliza un PC de uso doméstico con 4Gb de RAM, esto supone un problema de memoria y tiempo de cálculo. Por eso, se intentan varias soluciones para resolverlo:

Eliminación de bucles y valores innecesarios

Se revisa el proceso de obtención de los ángulos y se omiten bucles innecesarios reutilizando valores ya obtenidos para no tener que recalcularlos. Se eliminan operaciones elemento a elemento de cada matriz y se opera con matrices enteras.

Almacenamiento de los ángulos del vídeo

Se almacena la matriz de ángulos de todos los píxeles del vídeo en disco duro para evitar su almacenamiento en la memoria RAM, ralentizando el sistema y evitando recalcularla cada vez que se necesite. Sin embargo, la simple creación de la matriz produce un gran gasto de memoria ya que se va haciendo cada vez mayor en cada iteración con los frames, por lo que continuar con el cálculo lleva mucho más tiempo según se avanza. Si tras este período de tiempo, la memoria disponible es la suficiente y se ha logrado obtener, el simple hecho de guardarla en disco también consume más tiempo de lo deseado. Además, tras el posterior volcado de los datos para continuar con los cálculos, de nuevo nos encontraremos un problema de memoria.

Almacenamiento de los ángulos de los frames

Igual que en el punto anterior se procede a un volcado al disco de los ángulos de cada frame, esta vez uno por uno, evitando así utilizar mucha memoria RAM creando la matriz que agrupa los ángulos de todos los frames. Sin embargo, la continua operación de salvar estos

vectores conlleva mucho tiempo, y sólo nos servirán si los volcamos para hacer un cálculo secuencial porque si no, recurriremos de nuevo a un problema de memoria.

Valores para el cálculo secuencial

Tras estas aproximaciones, se decide llevar a cabo un cálculo secuencial, únicamente almacenando números únicos, y no la matriz completa. Esto es, en cada frame almacenar un valor necesario para que, al tener todos, se puedan seguir calculando la media y la desviación típica obteniendo un valor igual al que habríamos obtenido de haberlo hecho como se propuso inicialmente.

Para poder obtener los descriptores, debemos almacenar los datos adecuados para cada frame. En el caso de calcular la media y la desviación estándar se necesitan dos vectores, cada uno de ellos con una longitud de $n - 1$, siendo n el número de frames. Uno debe contener en cada una de sus posiciones las sumas de los cosenos de todos los ángulos de cada fotograma, y el otro las de los senos. Además, para calcular la desviación estándar también se necesitará ir almacenando el número de ángulos que se han tenido en cuenta en cada fotograma, ya que este no será constante a lo largo del vídeo debido al filtro de textura basado en la entropía aplicado en el apartado 4.3. Esto también será útil para calcular el módulo medio de cada fotograma, para el que también se necesitará almacenar la suma de todos los módulos de los vectores de movimiento para cada frame correspondiente, obteniendo también un vector de longitud equivalente a $n - 1$.

6.1.3. Extracción de descriptores estadísticos

Una vez aplicado el cálculo secuencial se procede a obtener los descriptores estadísticos a nivel de vídeo, resumidos en la tabla 6.1.

- **meanMods**: una vez obtenidos los vectores con la suma de los módulos de cada frame, éstos se suman a su vez y se divide entre el número de módulos tenidos en cuenta, obteniendo así el módulo medio del vídeo. Este descriptor dará una medida de

meanMods	Media del módulo de los vectores de movimiento del vídeo
modeCard	Punto cardinal moda de todos los frames
meanAngle	Ángulo medio de los vectores del vídeo
stdAngle	Desviación típica de los ángulos del vídeo
meanCard	Punto cardinal correspondiente al ángulo medio

Tabla 6.1: Descriptores estadísticos y su significado

la brusquedad o sutileza del movimiento medio del vídeo, ya que se relaciona con la velocidad de movimiento de los píxeles.

- **modeCard**: tras obtener el vector de direcciones nominales moda en cada frame, se obtiene la moda de éste, obteniendo así un número que representa la dirección más presente en todos los fotogramas del vídeo.

- **meanAngle**: se obtiene sumando todos los valores de los vectores c y s de cada frame, que almacenan la suma de cosenos y senos de todos los ángulos respectivamente. Después se procede al cálculo del ángulo medio como habríamos hecho a nivel de frame (ver fórmulas 6.2 y 6.3). Este descriptor nos da un único número, que representará el ángulo medio de todo el vídeo, expresado en grados, lo que nos proporcionará una idea de la dirección global de los vídeos.

- **stdAngle**: al sumar c y s, dividirlo por el número de píxeles válidos de todo el vídeo y hacer el módulo, obtenemos lo que sería $\bar{R}$ de todo el vídeo. Posteriormente sólo es necesario aplicar la fórmula 6.4 para obtener la desviación estándar del vídeo, que será un único número entre 0 e ∞. Esto da la variación de las direcciones de cada píxel respecto de la media, lo indica si el movimiento de cada uno de ellos es variado o, de lo contrario, monótono.

- **meanCard**: tras obtener el ángulo medio del vídeo, se obtiene su correspondencia con los puntos cardinales referidos en el apartado 5.1.3. Aporta la información del ángulo medio, sólo que de manera más interpretable.

6.2. Descriptores de movimiento de cámara

Tras haber calculado el tipo de movimiento de cámara presente en cada frame de los vídeos, sólo nos queda obtener representaciones numéricas de estos valores. Dado que son nominales, no podemos hacer cálculos estadísticos con ellos, pero sí se pueden calcular porcentajes a nivel de frame y a nivel de vídeo, así como modas. Estos descriptores están resumidos en la tabla 6.2.

modeTypeFrames	Movimiento moda de entre todos los frames
modeTypeShots	Movimiento moda de entre todos los planos
perShotFix	Porcentaje de planos con cámara fija
perFrameFix	Porcentaje de frames con cámara fija
perShotNo	Porcentaje de planos sin movimiento específico
perFrameNo	Porcentaje de frames sin movimiento específico
perShotPan	Porcentaje de planos con panning
perFramePan	Porcentaje de frames con panning
perShotTilt	Porcentaje de planos con tilt
perFrameTilt	Porcentaje de frames con tilt
perShotPanLeft	Porcentaje de planos con panning a la izquierda
perFramePanLeft	Porcentaje de frames con panning a la izquierda
perShotPanRight	Porcentaje de planos con panning a la derecha
perFramePanRight	Porcentaje de frames con panning a la derecha
perShotTiltUp	Porcentaje de planos con tilt hacia arriba
perFrameTiltUp	Porcentaje de frames con tilt hacia arriba
perShotTiltDown	Porcentaje de planos con tilt hacia abajo
perFrameTiltDown	Porcentaje de frames con tilt hacia abajo
perShotZoomIn	Porcentaje de planos con zoom in
perFrameZoomIn	Porcentaje de frames con zoom in
perShotZoomOut	Porcentaje de planos con zoom out
perFrameZoomOut	Porcentaje de frames con zoom out

Tabla 6.2: Descriptores de movimiento de cámara y su significado

- **modeTypeFrames** y **modeTypeShots**: se obtienen los movimientos de cámara más predominantes a nivel de frame y de plano. Aunque el segundo puede ser más relevante debido a que un plano es más perceptible visualmente, a nivel de frame se proporciona más detalle que quizá se podría perder a la hora de extrapolar la información a nivel de plano.

- **perShotFix/No/Pan/Tilt**: estos descriptores proporcionan el porcentaje de planos con cada tipo distinto de movimiento de cámara más observado, lo que excluye el zoom. Aporta en qué medida está cada tipo de movimiento presente o no en cada vídeo a nivel de plano, ya que quizá es útil saber la proporción de aquellos movimientos que no son predominantes.

- **perFrameFix/No/Pan/Tilt**: se aplican los mismos cálculos pero a nivel de frame, ya que puede ser que los umbrales establecidos para los movimientos de plano hagan perder información relevante a nivel de fotograma.

- **perShotPanLeft/PanRight/TiltUp/TiltDown/ ZoomIn/ZoomOut**: se especifica la dirección de cada tipo de movimiento a nivel de fotograma, para poder conocer si influye más la dirección de la cámara que el tipo de movimiento en sí. Debido a lo específico de estos descriptores, se incluye también el zoom, excluido previamente por su poca presencia.

- **perFramePanLeft/PanRight/TiltUp/ TiltDown/ZoomIn/ZoomOut**: de nuevo se llevan a cabo las mismas medidas, pero a nivel de frame.

7. Aprendizaje máquina

Gracias al trabajo previo realizado, se han obtenido 27 descriptores de los 138 vídeos de la base de datos, algunos relacionados con el ángulo y módulo de los vectores de movimiento, y la mayoría con el movimiento de cámara. Dado que la finalidad de éstos es ver su influencia a la hora de determinar la estética de los vídeos, el siguiente paso es llevar a cabo un proceso de aprendizaje máquina.

7.1. Weka

Para realizar un análisis del potencial de los descriptores para modelar la estética o percepción subjetiva de los vídeos, es necesario utilizar una herramienta *Weka*. *Weka* es un software *open source* en Java que aplica algoritmos de aprendizaje máquina y posee diferentes herramientas para llevar a cabo procesos de *clustering*, regresión, clasificación y preprocesado entre otras. *Weka* trabaja con diferentes tipos de atributos: numéricos, nominales, *strings*, fechas y de relación. Además tiene un tipo concreto de archivo con extensión *.arff* en cuya cabecera se indicarán los nombres de los atributos, así como su tipo, y posteriormente en cada fila sus valores en orden para cada una de las observaciones, como se observa en el siguiente ejemplo provisto en [17]:

```
1 @relation weather
2
3 @attribute outlook {sunny, overcast, rainy}
4 @attribute temperature real
5 @attribute humidity real
6 @attribute windy {TRUE, FALSE}
7 @attribute play {yes, no}
```

```
8
9 @data
10 sunny,85,85,FALSE,no
11 sunny,80,90,TRUE,no
12 overcast,83,86,FALSE,yes
13 rainy,70,96,FALSE,yes
14 rainy,68,80,FALSE,yes
15 rainy,65,70,TRUE,no
16 overcast,64,65,TRUE,yes
17 sunny,72,95,FALSE,no
18 sunny,69,70,FALSE,yes
19 rainy,75,80,FALSE,yes
20 sunny,75,70,TRUE,yes
21 overcast,72,90,TRUE,yes
22 overcast,81,75,FALSE,yes
23 rainy,71,91,TRUE,no
```

En este trabajo utilizaremos las funciones *Explorer*, para observar la tasa de aciertos de cada combinación de etiquetas y atributos, y *Experimenter* para, a su vez, comparar todos los resultados entre ellos y obtener aquellos más relevantes.

En este caso utilizaremos etiquetados de los vídeos basados en los gustos del usuario para poder determinar la influencia de cada descriptor en ellos. Como bien se refirió en el apartado 3, la base de datos utilizada para este trabajo es la misma que la que encontramos en [7]. Para este artículo se llevó a cabo el proceso de etiquetado de los diferentes vídeos según criterios basados en los metadatos que la plataforma *YouTube* provee con cada vídeo y aplicando la función de *clustering* que proporciona *Weka*. En este caso utilizaremos tres de estas clasificaciones: calidad -relacionada con los *likes* y *dislikes* del vídeo-, cantidad -basada en el número de visitas- y una combinación de ambas.

Antes de proceder a utilizar Weka, es necesitamos especificar cuáles de nuestros descriptores tendrán valor nominal y cuáles numérico, así como la clase a la que pertenecen. Denominaremos como nominal los descriptores moda y aquellos que sean referidos a números correspondientes a los cardinales: *modeTypeFrames, modeTypeShots,*

modeCard, meanCard y la clase a la que pertenecen. Por lo contrario, el resto de los descriptores serán de tipo numérico.

Para proceder a la clasificación será necesario utilizar el filtro de normalización para todos los atributos y poder trabajar a una misma escala. Las comparaciones de funcionamiento serán siempre basadas en el clasificador *ZeroR*, que nos sirve como referencia por ser el clasificador más simple, utilizando el método *cross-validation* con 10 hojas o *folds*.

7.2. Calidad

Para determinar el funcionamiento de un conjunto de atributos, cuyos mejores resultados están resumidos en 7.1, primero determinamos la tasa de acierto del clasificador *ZeroR*, que se sitúa en un 49.2454 %.

1. Se utilizan todos los descriptores y el clasificador con mejor resultado es *SimpleCart* que pertenece al grupo de árboles o *trees* y obtiene un 56.5217 % de acierto.

2. Posteriormente se eliminan todos aquellos atributos que no sean de carácter estadístico, y sólo se tienen en cuenta *meanCard, modeCard, meanAngle, stdAngle, modeTypeFrames, meanMods* y *modeTypeShots*. De nuevo el mejor clasificador es *SimpleCart* con un 57.2464 % de acierto.

3. Debido a la limitación de algunos algoritmos de selección de atributos en lo que respecta a los nominales, se eliminan *meanCard, modeCard, modeTypeFrames* y *modeTypeShots*. Tras esto se utiliza el algoritmo *SVMAttributEval* con *cross-validation* de 10 hojas. Con ello seleccionamos los 7 primeros atributos elegidos: *perFramePanRight, perFrameNo, perFrameTiltDown, perFrameZoomIn, meanMods, perShotZoomOut* y *perShotPanLeft*. Tras esto, los clasificadores *SMO (functions)* y *NaiveBayesMultinomialUpdateable* dan ambos un 59.42 % de acierto.

4. Se seleccionan diferentes agrupaciones de porcentajes de movimiento de cámara:

- Aquellos relacionados únicamente con el porcentaje de cada movimiento a nivel de plano, sin incluir las direcciones de estos: *perShotPan, perShotTilt, perShotNo* y *perShotFix*.
- De la misma manera se hará otra agrupación, pero a nivel de frame: *perFramePan, perFrameTilt, perFrameNo, perFrameFix*.
- Posteriormente se agruparán aquellos porcentajes que incluyen las direcciones del movimiento a nivel de plano: *perShotPanLeft, perShotPanRight, perShotTiltUp, perShotTiltDown, perShotZoomIn, perShotZoomOut, perShotNo* y *perShotFix*.
- Repetimos el proceso a nivel de frame: *perFramePanLeft, perFramePanRight, perFrameTiltUp, perFrameTiltDown, perFrameZoomIn, perFrameZoomOut, perFrameNo* y *perFrameFix*.

Los mejores resultados se obtienen con *RandomForest (tree)* en el caso de la primera agrupación y en el *SMO (functions)* de la tercera, siendo ambos de un porcentaje de acierto del 57.2464 %.

Conjunto atr.	Clasificador	Acierto
1	SimpleCart	56.5217 %
2	SimpleCart	57.2464 %
3	SMO/NaivesBayesMultUp	59.42 %
4	RandomForest/SMO	57.2464 %

Tabla 7.1: Atributos y clasificadores con mejor resultado en calidad

7.3. Cantidad

Respecto a las pruebas con las etiquetas de cantidad, se seguirá el mismo proceso de selección de atributos que en el caso de calidad. Por ello omitiremos en este caso la elección de atributos, ya que está previamente aludida en el apartado anterior. Los mejores resultados del análisis están resumidos en la tabla 7.2.

1. En el caso de la clasificación con todos los atributos, los mejores resultados obtenidos son con el clasificador *J48 (tree)*, que provee un acierto del 62.3188 %.

2. Tras tener en cuenta todos aquellos atributos relacionados con la media, la moda y la varianza, el mejor resultado lo proporciona *SimpleCart (tree)* con un 60.1449 % de acierto.

3. Tras eliminar nominales y hacer selección de atributos, se escogen *meanMods, perShotTilt, perZoomIn, meanAngle, perFrameZoomOut, perShotZoomOut* y *perFramePanLeft*. Los mejores resultados se obtienen de nuevo con *SimpleCart (trees)* y con *ComplementNaiveBayes*, donde en ambos casos el resultado es de un 59.4203 % de acierto.

4. En el caso de las diferentes agrupaciones según el porcentaje de frames y planos con cierto movimiento de cámara, los pocos casos que superan en funcionamiento al ZeroR, no lo hacen de manera destacable.

Conjunto atr.	Clasificador	Acierto
1	J48	62.3188 %
2	SimpleCart	60.1449 %
3	SimpleCart/ComplementNaiveBayes	59.4203 %

Tabla 7.2: Atributos y clasificadores con mejor resultado en cantidad

7.4. Combinación

Debido a que en este caso existen cuatro tipos diferentes de etiquetas, el valor de acierto del clasificador *ZeroR* es del 31.1594 %. El mejor resultado se puede ver en la tabla 7.3.

Los resultados absolutos respecto al porcentaje de acierto son notablemente inferiores a los casos de etiquetado binario. Esto es debido a que al manipular 4 etiquetas diferentes, la clasificación será más difícil y por ello los no es un caso comparable a los previos, aunque igualmente se pueden obtener resultados estadísticamente significativos.

En este caso se han probado las mismas combinaciones de atributos que en los casos previos y a la hora de llevar a cabo una selección con el algoritmo *SVMAttributeEval*, este conlleva mucho tiempo. Se han probado otros algoritmos pero igualmente sus resultados no son superiores al caso *ZeroR*. Sin embargo, en la utilización de valores únicamente relacionados con moda, media y desviación típica, se obtiene un 42.029 % de acierto al usar el clasificador *SimpleLogistic (functions)* y tras la eliminación de los valores nominales, un valor del 39.1304 % en el caso de *SimpleCart (tree)*.

Conjunto atr.	Clasificador	Acierto
3	SimpleLogistic	42.029 %

Tabla 7.3: Atributos y clasificador con mejor resultado en combinación

7.5. Comparación

Tras las diferentes clasificaciones realizadas con los tres etiquetados, se procede a comparar diferentes conjuntos de clasificadores y atributos mediante la herramienta *Experimenter* de *Weka*. Esto es necesario ya que aunque los porcentajes de acierto en algunos casos son más altos, a veces no implican una diferencia estadísticamente significativa. Se emplean los conjuntos mostrados en la tabla 7.4.

1	Todo el conjunto de atributos
2	meanAngle, meanMods, stdAngle, modeTypeFrames, modeTypeShots, meanCard
3	perFrameFix, perFrameNo, perFramePanLeft, perFramePanRight, perFrameTiltDown, perFrameTiltUp, perFrameZoomIn, perFrameZoomOut
4	perShotFix, perShotNo, perShotPanLeft, perShotPanRight, perShotTiltDown, perShotTiltUp, perShotZoomIn, perShotZoomOut

Tabla 7.4: Conjuntos de atributos aplicados en la comparación

En la tabla 7.5 se resumen los resultados estadísticamente significativos, así como su porcentaje de acierto, el etiquetado al que pertenecen, el clasificador y el tipo de conjunto de atributos cuyas asignaciones se corresponden con aquellos en la tabla 7.4.

Etiquetado	Conjunto atr.	Clasificador	Acierto
Combinación	1	SimpleLogic	38.12 %
Combinación	1	SimpleCart	40.22 %
Combinación	2	SimpleLogic	38.94 %
Combinación	2	SimpleCart	39.21 %
Cantidad	2	SimpleCart	56.90 %

Tabla 7.5: Resultados estadísticamente significativos

En el caso de las etiquetas combinadas, los resultados favorables pertenecen en la primera fila al conjunto entero de los atributos, y en la segunda al conjunto número 1 indicado en la tabla 7.4. Respecto a las etiquetas de cantidad, el resultado pertenece de nuevo al conjunto número 1 de la tabla 7.4.

7.6. Análisis de los resultados

Como se ha comentado previamente, los porcentajes de acierto en el caso del etiquetado combinado no son los valores más altos debido a la utilización de 4 clases de etiquetas en lugar de 2. Aun así, se observa que el etiquetado que más resultados estadísticamente significativos proporciona es el de combinación, ya que su mejora con respecto al *ZeroR* es tal que no se considera provocada por algo casual. Observamos que en este caso ayudan a su decisión o todos los atributos elegidos o aquellos relacionados con medidas estadísticas de ángulos y módulos, así como los movimientos de cámara más presentes a lo largo de los vídeos. Además, los algoritmos de clasificación que mejor funcionan son *Simple Logistic* y *SimpleCart*, siendo este último el que más casos signifiativos da de entre todas las combinaciones de etiquetado y atributos.

De esta manera, se puede concluir que aquellos descriptores más influyentes son aquellos relacionados con la dirección media del movmiento

8. Gestión del Proyecto

8.1. Etapas de realización

El proyecto se desglosa en pequeños hitos que a su vez engloban diferentes tareas con diferentes tiempos de realización.

Como etapa previa al comienzo se requiere buscar el trabajo a realizar, así como formalizar la asignación del trabajo de fin de grado y comenzar la documentación de su temática (ver tabla 8.1). En este caso no se añaden las horas aportadas por la búsqueda del proyecto y su asignación, ya que no suponen tiempo de trabajo en sí, aunque retrasan el comienzo de éste. Una vez se ha finalizado la documentación básica sobre el tema a tratar, se plantean pequeños hitos relacionados con el diseño del proyecto (ver tabla 8.2).

Fase	Tarea	Tiempo
Búsqueda de proyecto	Selección del TFG entre los disponibles	01/11/14 - 01/12/14
Asignación	Contacto con el tutor y asignación oficial	02/12/14 - 08/12/14
Documentación básica	Recopilación de documentación y estado del arte	21/01/15 - 03/02/15 (40 horas)
Total		40 horas

Tabla 8.1: Etapa preliminar

Fase	Tarea	Tiempo
Planificación	Se establece el calendario a seguir y sus hitos	04/02/2015 - 10/02/2015 (10 horas)
Diseño	Determinación de las técnicas a utilizar en cada hito	11/02/2015 - 24/02/2015 (20 horas)
Total		30 horas

Tabla 8.2: Etapa de diseño y planificación

Posteriormente se necesita probar el funcionamiento del algoritmo inicial seleccionado para la obtención de los primeros datos (ver tabla 8.3). La extracción de datos, aunque retrasa el tiempo del proyecto, no implica trabajo del autor ya que es realizado automáticamente. Una vez obtenidos los datos de partida, se procede a extraer las diferentes características de movimiento y a la extracción de descriptores a raíz de ellas (ver tabla 8.4).

Fase	Tarea	Tiempo
Pruebas flujo óptico	Se observa el funcionamiento del algoritmo de estimación de flujo óptico	08/04/2015 - 14/04/2015 (10 horas)
Selección de parámetros	Se determinan los parámetros a utilizar a lo largo del proyecto	08/04/2015 - 21/04/2015 (30 horas)
Extracción de datos	Se calculan los datos a raíz del algoritmo	06/05/2015 - 12/05/2015
Total		40 horas

Tabla 8.3: Etapa de prueba y extracción de datos

Fase	Tarea	Tiempo
Filtrado baja textura	Se establece un criterio para eliminar regiones homogéneas	13/05/2015 - 19/05/2015 (20 horas)
Cálculo de módulos	Se extraen los módulos del movimiento de los píxeles	24/06/2015 - 26/06/2015 (4 horas)
Cálculo de ángulos	Se extraen los ángulos del movimiento de los píxeles	24/06/2015 - 30/06/2015 (16 horas)
Detección de movimiento de cámara	Se desarrolla el proceso de detección de movimiento de cámara	01/07/2015 - 21/07/2015 (45 horas)
Pruebas	Se comprueba el correcto funcionamiento de ambos cálculos	22/07/2015 - 24/07/2015 (12 horas)
Total		97 horas

Tabla 8.4: Etapa de obtención de características

Una vez obtenidas las características, se procede a la extracción de los decriptores a nivel de vídeo así como a sus pruebas, cuyas fases se explican en la tabla 8.5. Además se procede, según se ve en la tabla 8.6, a la redacción de la memoria.

Se representan todas las actividades en un diagrama de Gantt, el cual por motivos de espacio ha sido dividido en cuatro partes (figuras 8.1, 8.2, 8.3 y 8.4) para su representación, omitiendo en gran parte aquellos retrasos notables en el comienzo de las actividades. Estos retrasos han sido debidos a períodos de exámenes, así como

Fase	Tarea	Tiempo
Obtención de valores para el cálculo secuencial	Se obtienen valores necesario para el cálculo de los descriptores de vídeo	17/08/2015 - 21/08/2015 (20 horas)
Extracción de descriptores	Se calculan todos los descriptores a partir de los valores previos	24/08/2015
Pruebas	Se estudia el valor de los descriptores en la evaluación estética	25/08/2015 - 31/08/2015 (20 horas)
Total		40 horas

Tabla 8.5: Etapa de extracción y análisis de descriptores

Fase	Tarea	Tiempo
Redacción de la memoria	Se expone el procedimiento llevado a cabo para el desarrollo del trabajo	08/07/2015 - 08/09/2015 (100 horas)
Corrección	Se corrigen fallos de estructura y redacción	09/09/2015 - 15/09/2015 (20 horas)
Total		120 horas

Tabla 8.6: Etapa de redacción de la memoria

vacacionales, en los cuales el tiempo disponible era más reducido y la coordinación con el tutor más complicada.

8.2. Presupuesto

El coste total del proyecto se resume en la tabla 8.7 y asciende a 8402,65€. Se desarrollará de manera más extensa en los próximos apartados.

Recursos	Coste
Personales	7530,84€
Software	69,00€
Equipo	388,81€
Otros	414,00€
Total	8402,65€

Tabla 8.7: Resumen de costes totales del proyecto

8.2.1. Coste personal

En la tabla 8.8 se representan las horas empleadas en cada etapa del proyecto, así como el total del tiempo empleado en su totalidad.

En base a esto, se obtiene el coste personal en relación a las horas invertidas y el salario por hora (ver tabla 8.9).

Figura 8.1: Diagrama de Gantt (1)

Etapa	Tiempo (horas)
Preliminar	40
Diseño y planificación	30
Prueba y extracción datos	40
Obtención de características	97
Extracción y análisis descriptores	40
Redacción de memoria	120
Total	367

Tabla 8.8: Tiempo total empleado

Recurso	Número de horas	€/hora	Coste total
Paloma Tirado Martín	367	20,52 [1]	7530,84€

Tabla 8.9: Tiempo total empleado

8.2.2. Costes de sofware

Para la realización de este proyecto se han utilizado cuatro tipos de software, los cuales se especifican en la tabla 8.10. El único gasto es debido a la obtención de la licencia oficial de *Matlab*, dado que el resto de softwares utilizados son *Open Source* y, por lo tanto,

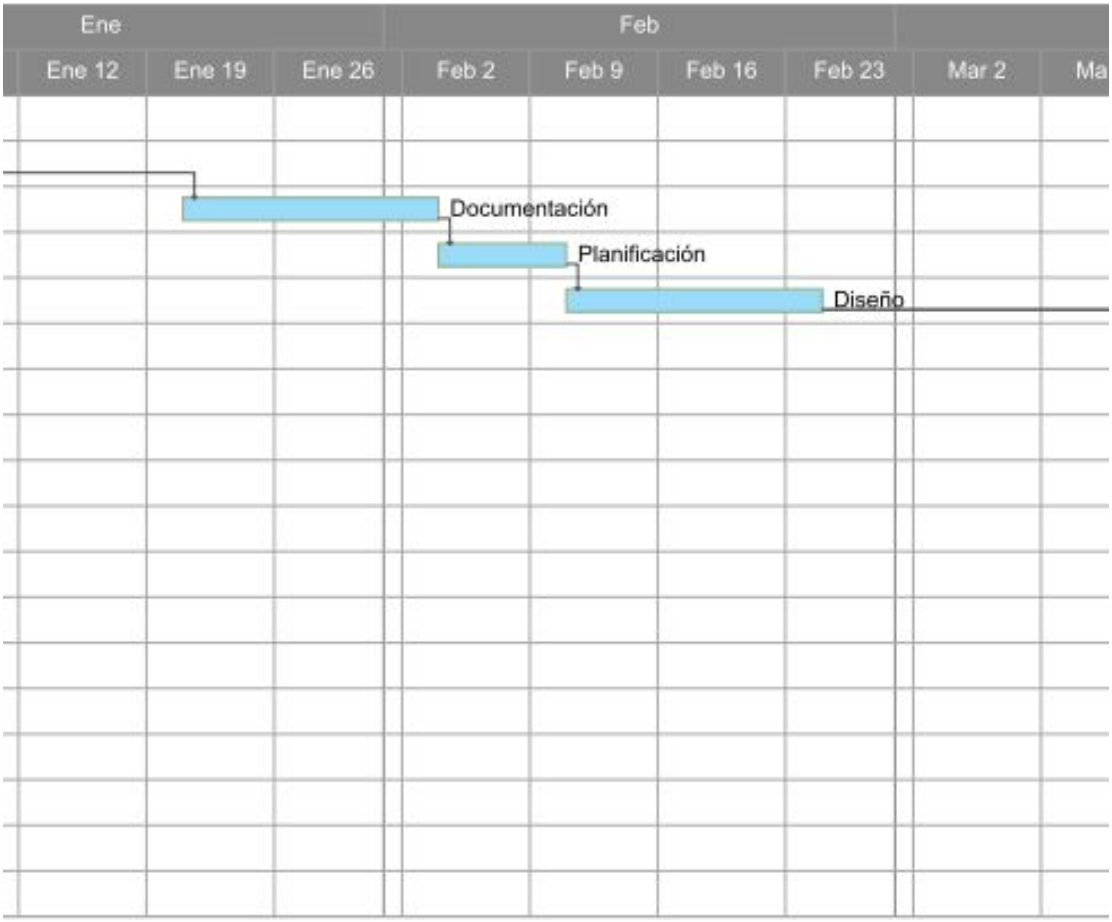


Figura 8.2: Diagrama de Gantt (2)

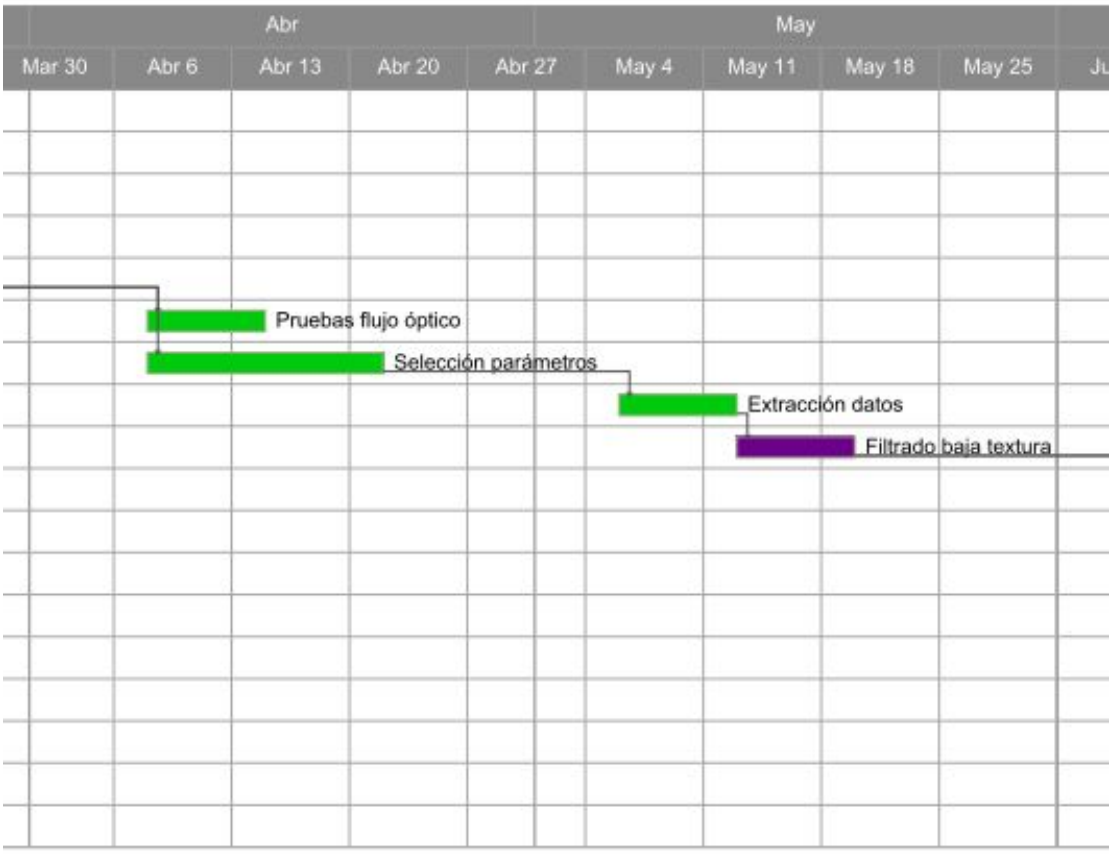


Figura 8.3: Diagrama de Gantt (3)

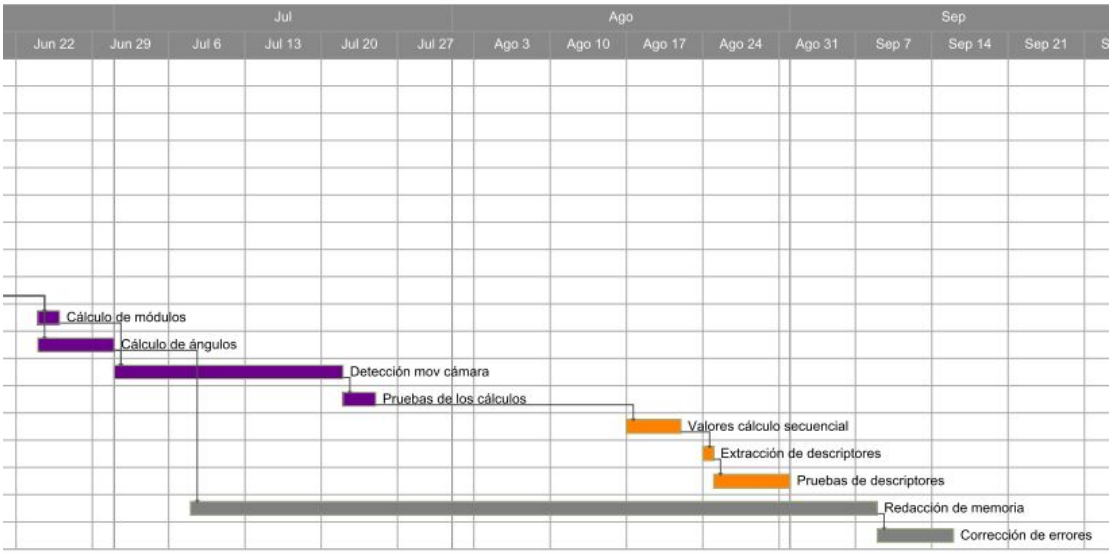


Figura 8.4: Diagrama de Gantt (4)

gratuitos. Esto produce un coste de 69,00€.

Recurso	Utilidad	Precio
MATLAB Student Suite 2014b	Empleado para la extracción de los descriptores	69,00 €
Weka 3.6	Software de aprendizaje máquina para la evaluación de los descriptores	Open Source - 0 €
TeXstudio	Editor de LaTeXpara la redacción de la memoria	Open Source - 0 €
Inkscape 0.91	Software para la creación de gráficos	Open Source - 0 €
Total		69,00 €

Tabla 8.10: Coste en relación al software utilizado

8.2.3. Costes de equipo

En cuanto al equipo utilizado, se tendrá en cuenta un periodo de depreciación de 60 meses, produciendo un coste de 388,81€, como indica la tabla 8.11.

Recurso	Coste	% de uso	Depreciación	Coste de atribución
Toshiba Satellite L755	699,00€	100	60	372,8€
hama optical mouse AM100	10,00€	100	60	2€
C3P0 Keyboard KBR36	29,00€	100	60	14,01€
Total				388,81€

Tabla 8.11: Coste en relación al equipo utilizado

8.2.4. Otros costes directos

Así, en relación a otro tipo de costes, se encuentan aquellos relacionados con el servicio de ADSL necesario para la conexión a Internet dado que se requiere búsqueda de documentación (ver 8.12). Esto se resumirá en unos costes de 414,0€.

Descripción	Empresa	Coste
Conexión ADSL 10 MB (10 meses)	Telefónica S.A.	41,40€/mes × 10 meses = 414,0€
Total		414,0€

Tabla 8.12: Coste sobre otros servicios requeridos

9. Conclusions and future work

This chapter will finish this document, summarizing all the conclusions extracted from the work explained on it and proposing future improvements on aesthetic assessment.

9.1. Conclusion

Assessing videos' aesthetic automatically is not trivial due to subjectivity related to humans' likings. Database labeling, even being manual can be ambiguous depending on the person who does it. This is why carrying out this task automatically is even more difficult.

The three labeling types used in this project are not special as they are obtained through metadata provided by *YouTube*, which do not represent every single viewer's perception. Based on the number of views, this does not always imply that the video is appealing to every user. The number of times users have pressed the like or dislike buttons does not represent with accuracy de amount of users who liked the video or not, because not all of them decided to press one of these buttons. Because of this we will not have a ground-truth fully in line with the perception of each person who has watched the video.

The lack of a correctly labeled database also affects when determining the camera motion of all the different videos. As we do not have a camera motion labeling on each different shot, it is difficult to check the correct performance of the proposed estimation method. Even though some of them are manually labeled, this is not entirely reliable in ambiguous situations. This is why, despite of testing the estimation in some of the videos, its efficiency on the remaining database is unknown.

Together this might induce that the obtained results after ma-

chine learning could not be correspondent to those which could be logical from an audiovisual point of view. Although it seems obvious that camera motion affects notably in video aesthetics, we do not get a great influence from these descriptors when differentiating them by their type of movement or direction. Despite this, we get more positive results with descriptors related to modulus, angle and the most present camera motion on the video. However, the percentage of each type of camera motion, at frame or shot level, are less determinant than the most present camera movement. After that, we can deduce that motion direction and velocity along the video affect the aesthetic appearance. However, having more or less presence of a specific type of camera motion does not affect in the same way as the general one, the most present.

9.2. Future work

Due to problems when establishing ground-truth about aesthetic appearance, a possible improvement would be carrying out a supervised labeling by people who evaluate their perception gradually. Despite of still having subjectivity problems, when having a gradual evaluation it would be easier to get labels more consistent with each user's sentiment.

Through the utilization of sensors which measure involuntary responses of the evaluators, such as heart rate or cerebral activity, a more reliable labeling could be done. Although there will still be subjectivity in people's likings, it would be a gradual and sincere respond because it would be related to factors that are not consciously controllable.

Since fixed cameras have been detected by calculating the modulus of optical flow vectors, this imply errors when detecting hand-carried cameras. Sometimes this type of camera motion is so subtle that is estimated as fixed as well. Due to this, it would be useful to add software in the future that would be capable of differentiating between these two types of camera motion since they have a strong presence in the database. An approach could be segmenting each frame, making possible to distinguish those regions which belong to the background. This way, fixed cameras would be detected more re-

liably, given that the vectors would be less than in the margins, and the mean modulus threshold could be much more discriminative, helping to discard hand-carried camera motion.

Bibliografía

[1] Presupuesto proyecto fin de carrera según la universidad carlos iii de madrid.

[2] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2010.

[3] Pere Obrador Anush K. Moorthy and Nuria Oliver. Towards computational models of the visual aesthetic appeal of consumer videos. In *Computer Vision ECCV 2010*, pages 1–14, 2010.

[4] Michael J. Black and Allan D. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996.

[5] B.F. Buxton and H. Buxton. Computation of optic flow from the motion of edge features in image sequences. *Image and Vision Computing*, 2(2):59–75, 1984.

[6] Javier Sánchez Enric Meinhardt-Llopis and Daniel Kondermann. Horn-schunck optical flow with a multiscale stategy. *Image Processing On Line*, 2013.

[7] A. Hernández García F. Fernández Martínez and F. Díaz de María. Succeeding metadata based annotation scheme and visual tips for the automatic assessment of video aesthetic quality in car commercials. *Expert Systems with Applications*, 42:293–305.

[8] B.K.P. Horn and B.G. Schunck. Determining optical flow. *AI Memo 572, Massachusetts Institute of Technology*, 1980.

[9] Ming-Sui Lee Hsin-Ho Yeh, Chun-Yu Yang and Chu-Song Chen. Video aesthetic quality assessment by temporal integration of photo- and motion-based features. *IEEE Transactions On Multimedia, Vol. 15, Nº. 8*, pages 1784–1791, 2013.

[10] S. Rao Jammalamadaka and A. SenGupta. *Topics in Circular Statistics*. World Scientific Publishing, New Jersey, USA, 2001.

[11] Ce Liu Jenny Yuen, Bryan Russell and Antonio Torralba. Labelme video: Building a video database with human annotations. In *IEEE 12th International Conference*, pages 1451–1458, 2009.

[12] D.J. Fleet J.L. Barron and S.S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[13] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Artificial Intelligence Review*, 26(3):159–190, 2006.

[14] Petra Krämer and Jenny Benois-Pineau. Camera motion detection in the rough indexing paradigm. In *Image Analysis and Recognition: 9th International Conference*, 2010.

[15] Yantao Zheng-Changsheng Xu Qi Tian Jesse S. Jin Ling-Yun Duan, Jinqiao Wang and Hanqing Lu. Shot-level camera motion estimation based on a parametric model. *TRECVID*, 2005.

[16] C. Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

[17] Locualo.net. Minería de datos con weka (ficheros arff), 2007.

[18] Diane Larlus Luca Marchesotti, Florent Perronnin and Gabriela Csurka. Assessing the aesthetic quality of photographs using generic image descriptors. In *IEEE International Conference on Computer Vision*, 2011.

[19] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of*

the 7th international joint conference on Artificial intelligence, volume 2, pages 674–679, 1981.

[20] S. Venkatesh M. V. Srinivasan and R. Hosie. Qualitative estimation of camera motion parameters from video sequences. *Pattern Recognition*, 30(4):593?606, 1997.

[21] T.M. Mitchell. *Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression.* McGraw Hill, 2015.

[22] Naila Murray and Florent Perronnin Luca Marchesotti. Ava: A large-scale database for aesthetic visual analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2408–2415, 2012.

[23] Olivier Nikos Paragios, Yunmei Chen and D. Faugeras. *Handbook of Mathematical Models in Computer Vision.* Springer Science+Business Media, Inc., New York, USA, 2006.

[24] Peter O'Donovan. Optical flow: Techniques and applications. 2005.

[25] J.P. Lewis Stefan Roth Michael J. Black Simon Baker, Daniel Scharstein and Richard Szeliski. A database and evaluation methodology for optical flow. *Eleventh IEEE International Conference on Computer Vision (ICCV 2007)*, October 2007.

[26] Tao Chen Dong Liu Shih-Fu Chang Subhabrata Bhattacharya, Behnaz Nojavanasghari and Mubarak Shah. Towards a comprehensive computational model for aesthetic assessment of videos. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 361–364, 2013.

[27] Richard Szeliski. Computer vision: Algorithms and applications. Draft.

[28] Nils Papenberg Thomas Brox, Andres Bruhn and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *8th European Conference on Computer Vision*, volume 4, pages 25–36, 2004.

[29] Roman Timofeev. Classification and regression trees (cart) theory and applications. Master's thesis, Humboldt University, Berlin, 2004.

[30] Cathleen Wei Jiang, Alexander C. Loui and Daniels Cerosaletti. Automatic aesthetic value assessment in photographic images. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 920–925, 2010.